Zeszyty Naukowe Wydziału Informatyki
Wyższej Szkoły Informatyki Stosowanej i Zarządzania
„Informatyka Stosowana”
Nr 1/2014

EWOLUCYJNE METODY ZNAJDOWANIA STRUKTUR TYPU „KERNEL & SHELL” W GRAFIE POŁĄCZEŃ

Jarosław Stańczak, Krzysztof Sęp

Instytut Badań Systemowych Polskiej Akademii Nauk,
01-447 Warszawa, ul. Newelska 6

Streszczenie

W pracy opisano ewolucyjne metody uzyskiwania struktury „kernel & shell” w grafie połączeń. Strukturę taką zaproponowali autorzy jako rozszerzenie znanej i używanej w zagadnieniach komunikacyjnych struktury „hub & spoke”, umożliwiającej przekształcanie nieuporządkowanej struktury grafu połączeń, do postaci ułatwiającej rozwiązywanie złożonych problemów komunikacyjnych. Zasadą tego typu przekształceń jest wyróżnienie w grafie grupy silnie powiązanych ze sobą węzłów (hubów), tworzących strukturę „kernel” – jądro oraz związanych z nimi lokalnie węzłów peryferyjnych, tworzących „shell”. Ponieważ zadanie to jest skomplikowane obliczeniowo, użyto do tego celu algorytmu ewolucyjnego w kilku wariantach, zależnie od możliwych wymagań rozpatrywanego problemu.

Słowa kluczowe: transport, graf połączeń, „hub & spoke”, „kernel & shell”, α-klika, algorytm ewolucyjny

1. Wstęp

Struktura „kernel & shell” (*kernel-and-shell* lub *kernel and shell*) w grafie połączeń jest rozszerzeniem znanej i używanej w zagadnieniach komunikacyjnych struktury „hub & spoke” (*hub-and-spoke* lub *hub and spoke*), umożliwiającej przekształcanie nieuporządkowanej struktury grafu połączeń, do postaci ułatwiającej rozwiązywanie złożonych problemów komunikacyjnych. Podstawową ideą tego typu przekształceń jest wyróżnienie w grafie grupy ściśle powiązanych ze sobą węzłów, tworzących strukturę „kernel” – jądro oraz związanych z nimi lokalnie węzłów peryferyjnych, tworzących „shell”. Można taki podział uznać również za swoisty klastering grafu, w którym to klaster tworzący kernel ma zawsze dokładnie jeden węzeł wspólny z każdym wydzielonym klastrem typu shell.

W zależności od wymagań i struktury grafu wymagane jest, aby węzły, tworzące „kernel”, stanowiły strukturę o jak najlepszej jakości i liczbie połączeń między węzłami. Idealny byłby w wielu przypadkach podgraf pełny, czyli klika. Jeśli taka idealna sytuacja jest niemożliwa (a tak jest w większości typowych przypadków), to

możemy wymagać, aby stanowiły one tzw. α-klikę, czyli, w uproszczeniu, niepełną klikę o określonych parametrach, zdefiniowaną w dalszej części pracy. W ostateczności, jeśli rozpatrywany graf jest bardzo rzadki, to wymaga się, aby „kernel" był przynajmniej podgrafem spójnym. W przypadku węzłów peryferyjnych, czyli węzłów typu „shell" wymagania mogą być bardziej skomplikowane, gdyż w zależności od rozpatrywanego problemu można zażądać aby:

- każdy węzeł peryferyjny był połączony ze swoim reprezentantem struktury „kernel", tzw. hubem, powstaje wówczas klasyczna struktura „hub and spoke";
- węzły peryferyjne tworzyły wraz ze swoim hubem klikę, lub, częściej, α-klikę, czyli strukturę w której istotne są również połączenia pomiędzy węzłami lokalnymi należącymi do jednego hub-a;
- zarówno dla elementów struktury „kernel" jak i węzłów lokalnych istotne mogą być nie tylko same fakty istnienia połączeń, ale również wagi tych połączeń.

Istniejące powiązania węzłów lokalnych z innymi hubami lub węzłami lokalnymi, należącymi do innych hubów są ignorowane i takie połączenia są tworzone jedynie za pośrednictwem odpowiednich hub-ów i sieci połączeń pomiędzy nimi.

Uwzględnienie wag połączeń i ewentualnie braku symetrii połączeń w grafie, to kolejne problemy, które mogą utrudniać rozwiązywanie tego typu zadań, i które są rozpatrywane w niniejszej pracy.

Struktury połączeń, podobne do wymienionych powyżej, dają wiele korzyści w stosunku do grafu o nieuporządkowanej strukturze. Ułatwiają projektowanie nowych sieci, umożliwiają lepsze wykorzystanie istniejących zasobów, przy odpowiednio przeprowadzonej reorganizacji, poprawiają jakość i częstotliwość połączeń oraz redukują ruch w sieci, bez pogorszenia parametrów połączeń. W związku z takimi zaletami tego typu sieci ważne jest stworzenie metod i narzędzi umożliwiających transformacje grafów połączeń do wybranych struktur z możliwością uwzględnienia różnych uwarunkowań rozpatrywanych problemów. Ponieważ są to zagadnienia o znacznej złożoności obliczeniowej, do których nie powstały jak dotąd efektywne metody specjalizowane, do ich rozwiązania wykorzystano odpowiednio przystosowane metody ewolucyjne. Niniejsza praca poświęcona jest w dużej części przedstawieniu tych ewolucyjnych metod przekształcania nieuporządkowanych grafów połączeń do opisanych struktur typu kernel and shell.

2. Pojęcia podstawowe

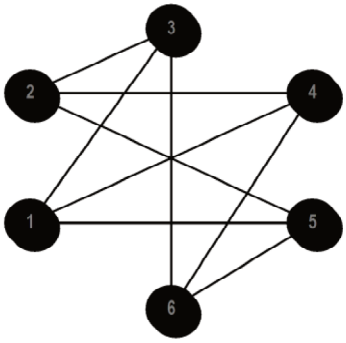
2.1 Podstawowe pojęcia związane z grafami

Graf jest parą $G=(V, E)$, gdzie V jest niepustym zbiorem *wierzchołków,* a E jest zbiorem *krawędzi*. Każda krawędź jest parą wierzchołków (v_1, v_2) takich, że $v_1 \neq v_2$. Dwa wierzchołki grafu $G=(V, E)$ są ***incydentne*** jeżeli $v_1,v_2 \in V$ to $\{v_1,v_2\} \in E$.

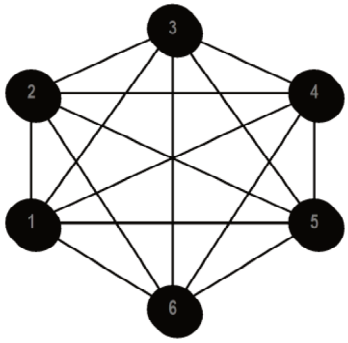
Podgraf grafu $G = (V, E)$ jest grafem $G' = (V', E')$, gdzie $V' \subseteq V$ i $E' \subseteq E$ takim, że dla każdego $e \in E$ i $e=\{v_1,v_2\}$ jeżeli $v_1,v_2 \in V'$ to $e \in E'$.

Drogą (ścieżką) w grafie $G=(V, E)$ z wierzchołka s do wierzchołka t nazywamy ciąg wierzchołków $\{v_1, \ldots, v_n\}$ taki, że: $\{s, v_1\} \in E$, $\{v_i, v_{i+1}\} \in E$ dla $n=1, 2, \ldots, n-1$, $\{v_n, t\} \in E$.

Graf $G=(V, E)$ jest ***grafem spójnym***, jeżeli dla każdych dwóch różnych wierzchołków istnieje droga łącząca te wierzchołki.

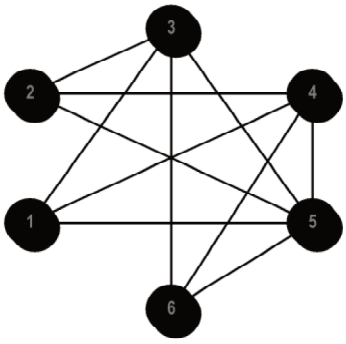


Rys. 2.1 Przykładowy graf

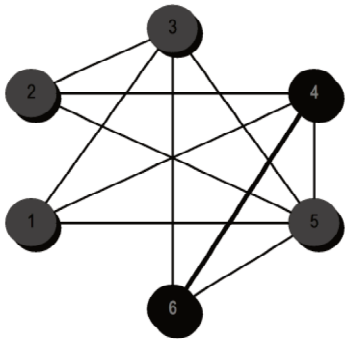


Rys. 2.2 Graf pełny

Kliką (*podgrafem pełnym*) $Q=(V_q, E_q)$ w grafie $G=(V, E)$ jest graf taki, że $V_q \subseteq V$ i $E_q \subseteq E$ oraz każda para wierzchołków $v_1, v_2 \in V_q$ spełnia warunek: $\{v_1, v_2\} \in E_q$.



Rys.2.3 Przykładowy graf



Rys. 2.4 Kliki w grafie z rys. 2.3.

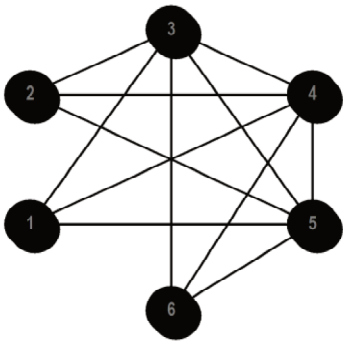
Na rysunku 2.4 mamy zaznaczone dwie kliki: {1, 3, 5} i klikę {2, 3, 5}.

Maksymalną kliką nazywamy klikę $Q_M=(V_q, E_q)$ w grafie $G=(V, E)$ taką, że nie istnieje wierzchołek $v \in V$ i $v \notin V_q$ taki, że $Q'=(V', E')$ jest kliką, gdzie $V'=V \cup \{v\}$ i $E' \subseteq E$ gdzie każda para $v_1, v_2 \in V'$ wierzchołków spełnia warunek $\{v_1, v_2\} \in E'$

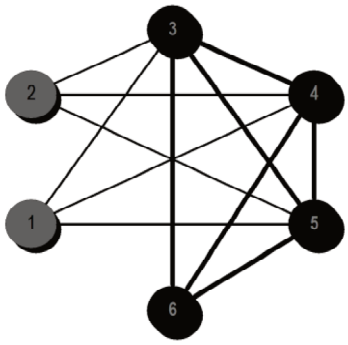
Stopniem wierzchołka w *grafie prostym* (*nieskierowanym*) nazywamy liczbę krawędzi, do których ten wierzchołek należy.

Na przykład, wierzchołek 2 w grafie z rysunku 2.5 ma stopień 3, a wierzchołek 4 ma stopień 5.

Macierz sąsiedztwa jest macierzą kwadratową o rozmiarze n równym liczbie węzłów w grafie. Każdy jej element, w ogólnym przypadku, pokazuje liczbę krawędzi pomiędzy dwoma dowolnymi wierzchołkami, z zerami na głównej przekątnej. Jest ona jedną z form matematycznego zapisu grafów i jest najczęściej wykorzystywaną formą zapisu informacji o grafie, używaną w tej pracy. W przypadku wykorzystywanych w pracy grafów prostych i nieskierowanych jest to macierz symetryczna, w której wartość 1 oznacza istnienie krawędzi, a 0 jej brak. Ponieważ w wykorzystywanych w pracy pojęciach kliki i α-kliki istotne jest sąsiedztwo wierzchołka samego ze sobą (pojedynczy wierzchołek jest α-kliką o α=1, czyli kliką), to zdecydowaliśmy się na umieszczanie 1 również na głównej przekątnej macierzy sąsiedztwa, co znacznie ułatwia obliczenia.



Rys. 2.5 Przykładowy graf



Rys. 2.6 Podgraf pełny grafu z rys. 2.5

2.2 α-klika

Niech $A=(V_\alpha, E_\alpha)$ będzie podgrafem grafu $G=(V, E)$, gdzie $\alpha\in(0,1\rangle$, $V_\alpha\subseteq V$, $E_\alpha\subseteq E$, $k=Card(V_\alpha)$, k_i jest liczbą wierzchołków $v_j\in V_\alpha$ takich, że $\{v_i, v_j\}\in E_\alpha$.

1. Dla k=1 podgraf A grafu G jest ***α-kliką(α)***.
2. Dla k>1 podgraf A grafu G jest ***α-kliką(α)***, jeżeli dla każdego $v_i\in V_\alpha$ zachodzi warunek:

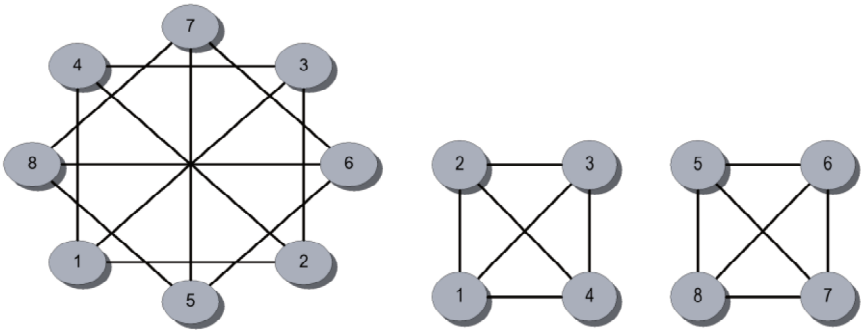
$$\alpha \le \frac{k_i + 1}{k} \tag{2.1}$$

Wdalszym ciągu, dla oznaczenia ***α-kliki(α)*** będziemy używali skróconego zapisu ***α-klika*** (lub alfaklika), rozumiejąc je dla ustalonego wcześniej ***α***.

Więcej informacji o właściwościach i możliwościach wykorzystania ***α-kliki*** można znaleźć w pracach Potrzebowskiego, Stańczaka i Sepa (2006a, b, 2007). Poniżej przedstawione zostaną niektóre właściwości ***α-kliki***.

1. Weźmy dowolny wierzchołek v_i, należący do α-kliki $A=(V_\alpha, E_\alpha)$.
   Niech $k=Card(V_\alpha)$, k_i jest liczbą wierzchołków $v_j\in V_\alpha$ takich, że $\{v_i,v_j\}\in E_\alpha$.
   Dla $\alpha>1/2$ mamy $k_i/k\ge\alpha>1/2$, czyli $k_i/k>1/2$, zatem $k_i>k/2$.

Z rachunku zbiorów wynika, że dla każdych dwóch wierzchołków zbiory wierzchołków z nimi incydentnych mają część wspólną, zatem graf jest spójny. Dla α=1/2 odpowiedni graf nie musi spełniać warunku spójności. I tak, na przykład, graf z rys. 2.7 nie jest spójny.

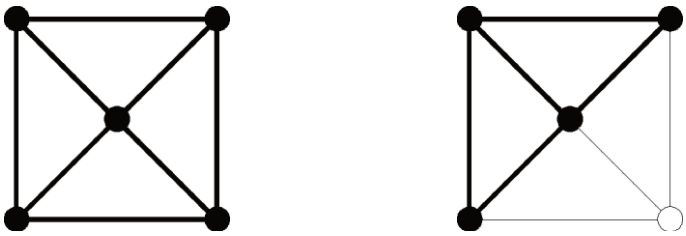


Rys 2.7 Przykład grafu niespójnego dla. $\alpha = \frac{1}{2}$

Zatem dla $\alpha > \frac{1}{2}$ α-klika jest grafem spójnym, czyli grafem, w którym dla każdych dwóch różnych wierzchołków istnieje ciąg krawędzi należących do grafu, łączących te wierzchołki.

2.Jeżeli w grafie istnieje k wierzchołkowa α-klika(α) to nie każdy podzbiór wierzchołków tej α-kliki musi być α-kliką(α).
Wynika to z zależności:

$$\frac{k}{n} < \frac{k+1}{n+1}$$



a) b)

Rys. 2.8 a) Graf G, 5 wierzchołkowy b) Graf G' 4 wierzchołkowy podgraf grafu G.

Graf G jest α-kliką dla $\alpha = \frac{4}{5}$ podczas gdy graf G' nie jest α-kliką dla $\alpha = \frac{4}{5}$ (rys. 2.8)

Algorytm zachłanny

Maksymalna klika

Niech:

$G(V, E)$ – graf, w którym V jest niepustym zbiorem wierzchołków, a E jest zbiorem krawędzi,
Q – podzbiór zbioru wierzchołków V (klika),
$deg(v)$ – stopień wierzchołka,
$maxdeg(G)$ – wierzchołek o największym stopniu w grafie G,
$remove(G, v)$ – usuwa z grafu G wierzchołek v oraz wszystkie krawędzie do których ten wierzchołek należy,
$Y(Q)$ – podzbiór zbioru wierzchołków V takich, że każdy wierzchołek z Y jest incydentny ze wszystkimi wierzchołkami z Q,
Z – graf w którym Y jest zbiorem wierzchołków, a zbiorem krawędzi jest podzbiór zbioru V takich, że każda krawędź jest parą wierzchołków (v_1, v_2) takich, że $v_1 \neq v_2$ oraz $v_1, v_2 \in Y$,
Cl – zbiór grafów $\{G_1(V_1, E_1), G_2(V_2, E_2), \ldots, G_m(V_m, E_m)\}$ takich, że $V_i \cap V_j = \varnothing$ dla każdego $i \neq j$ oraz $i, j = 1, 2, \ldots, m$, $V_1 \cup V_2 \cup \ldots \cup V_m = V$.

Maksymalna klika

```
procedure max_clique(G)
input
V;
E;
output
Q;
begin
Q:=∅;
Q= Q ∪ maxdeg(G);
        while Y(Q)≠∅ do
                begin
                    Q= Q ∪ maxdeg(Z);
                    remove(G, maxdeg(Z));
                end;
end;
```

Dla problemu maksymalnej α-kliki wystarczy, aby zbiór $Y(Q)$ był takim podzbiorem zbioru wierzchołków V, by po dodaniu do zbioru Q dowolnego wierzchołka ze zbioru $Y(Q)$, zbiór Q wraz z odpowiednimi krawędziami był α-kliką.

Aby uzyskać podział grafu $G(V, E)$ proponujemy następujący algorytm:

Klastering

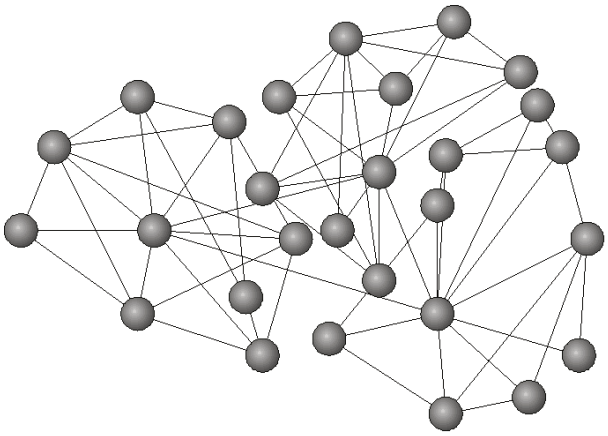
```
procedure cluster(G)
input
V;
E;
output
Cl;
begin
Cl:=∅;
        while V≠∅ do
                begin
                    Cl=max_clique(G); //Cl{Vi, Ei}
                    V=V\Vi;
                    E=E\Ei;
                end;
end;
```

Powyższy algorytm podaje jedno rozwiązanie, którego dalej nie modyfikuje. Algorytm ten jest algorytmem zachłannym, który podaje tylko aproksymację rozwiązania optymalnego.

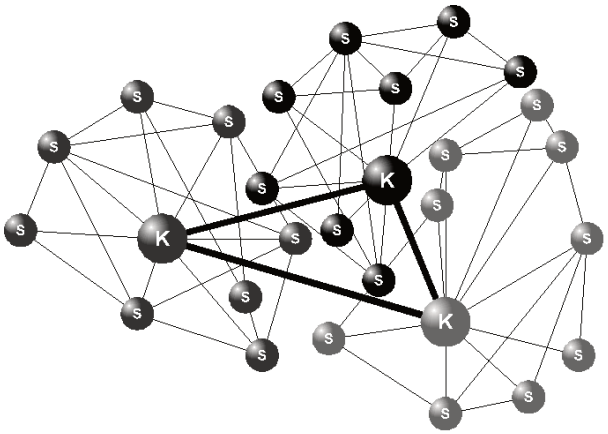
2.3 Kernel and shell

Graf *G(V, E)* złożony z dwóch podgrafów:

- grafu spójnego $K(V_k, E_k)$, będącego α-kliką o wartości α-ustalonej w zależności od specyfiki zadania. Graf ten będziemy nazywali – kernel.
- grafu $S(V_s, E_s)$ takiego, że $V_s = V - V_k$ i $E_s = E - E_k$, który będziemy nazywali shell.



Rys. 2.9. Struktura wejściowa.



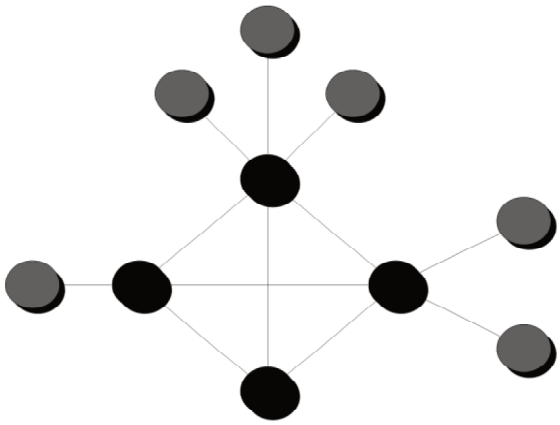
Rys. 2.10. Odpowiadająca strukturze wejściowej struktura kernel and shell.

Pierścieniem w strukturze kernel and shell generowanym przez wierzchołek z grafu kernel v_k będziemy nazywali zbiór wszystkich wierzchołków należących do grafu shell i połączonych z wierzchołkiem v_k.

Pierścienie nie muszą być rozłączne, ale rodzina wszystkich pierścieni pokrywa cały shell.

Szczególnym przypadkiem struktury kernel and shell jest struktura hub and spoke opisana przez O'Kelly'ego i Bryana (2002), a zdefiniowana przez Potrzebowskiego, Stańczaka i Sępa (2008).

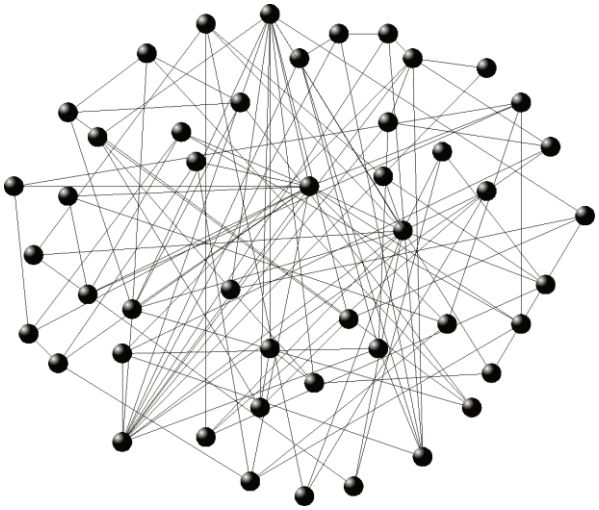
Struktura hub-and-spoke to graf $H_s=(V_h \cup V_s, E)$, w którym niepusty podzbiór V_h wraz z odpowiednimi krawędziami wyznacza graf pełny. Każdy wierzchołek z podzbioru V_s ma stopień 1 i jest połączony z dokładnie jednym wierzchołkiem ze zbioru V_h (Mażbic-Kulma i in., 2008).



Rys. 2.11. Przykład struktury typu hub-and-spoke.

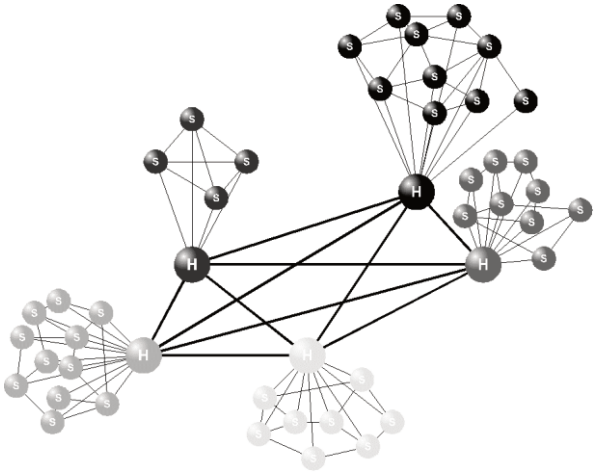
Struktura hub-and-spoke (HS) zdefiniowana powyżej może być zastosowana w wielu dziedzinach. Używając tej struktury w systemach transportowych (O'Kelly, 1987) uzyskujemy dużą koncentrację ruchu pomiędzy węzłami oraz synchronizację połączeń, a zatem zwiększamy jakość usług. Powoduje to również obniżenie kosztów przewozu pasażerów. Struktura hub-and-spoke została wykorzystana w transporcie miejskim we Frankfurcie nad Odrą. Wydzielono tam 4 węzły przesiadkowe (huby). Wprowadzenie hub-and-spoke spowodowało znaczne zwiększenie częstotliwości ruchu. Funkcjonalność tego systemu jest szczególnie widoczna w godzinach pozaszczytowych, gdy oferta przewozowa jest już z powodów ekonomicznych ograniczona.

Ze względu na to, że w rzeczywistych zadaniach trudno jest uzyskać strukturę hub and spoke, w dalszej części niniejszej pracy strukturą hub and spoke będziemy nazywali pojęcie rozszerzone, gdzie graf hubów jest α-kliką o $\alpha \leq 1$, czyli nie koniecznie kliką, a uzyskana struktura jest strukturą typu kernel and shell zbliżoną do struktury hub and spoke.

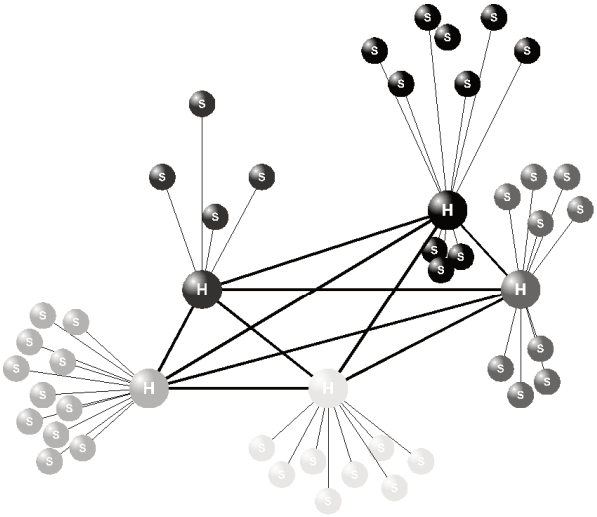


Rys. 2.12. Graf wejściowy.

Sruktura kernel and shell z rysunku 2.13 zawiera tylko rozłączne pierścienie. W niniejszej pracy będziemy się zajmowali wyłącznie strukturami o rozłącznych pierścieniach.



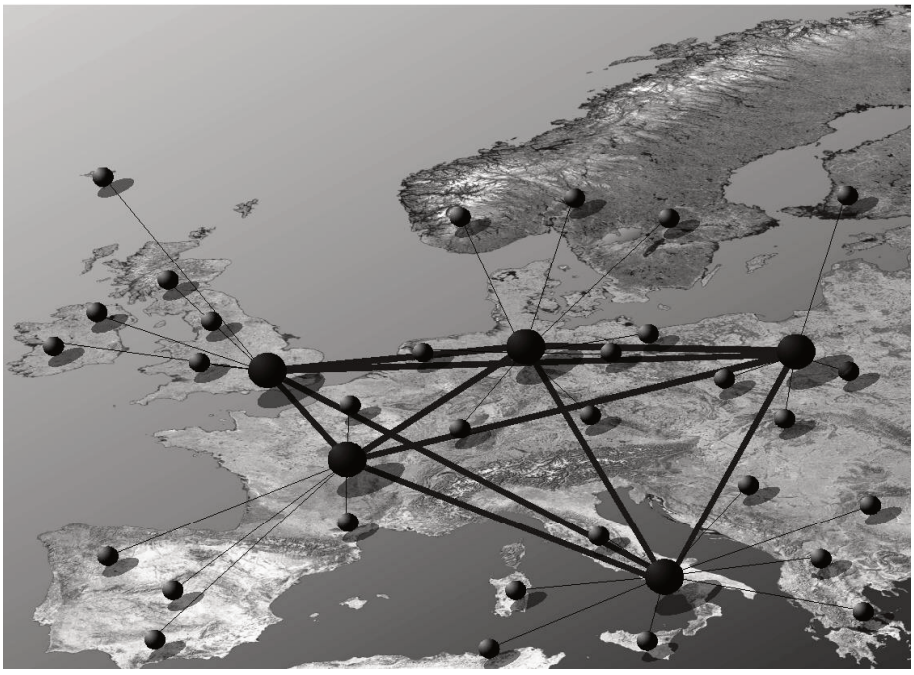
Rys. 2.13. Struktura kernel and shell dla grafu wejściowego z rys. 2.12.



Rys. 2.14. Struktura hub and spoke dla grafu wejściowego z rys. 2.12.

Przykładem może być optymalizacja połączeń w ruchu lotniczym.

Rys. 2.15. Grafowa struktura połączeń.



Rys. 2.16. Odpowiadająca Rys. 2.15 struktura typu hub and spoke.

3. Metoda „kernel and shell”

Metoda “kernel and shell” jest właściwie pewnym ogólnym schematem połączeń, strukturą połączeń, wiodącą do uproszczenia projektowanych połączeń w pewnej sieci. Zawiera ona wiele możliwości podziału grafu połączeń na podgrafy, tworzące kernel i shell, w zależności od konkretnych wymagań rozwiązywanego problemu. W niniejszej pracy rozpatrywane są metody mające pewną wspólną cechę. Cechą tą jest rozłączność podgrafów tworzących shell. O ile węzły tworzące kernel powinny być ze sobą ściśle połączone i stanowią rusztowanie grafu połączeń, to węzły podgrafu shell połączone są tylko ze swoim reprezentantem struktury kernel (zwanym często hubem) i ewentualnie innymi węzłami tego samego shella. Założenie o rozłączności struktur shell nie jest aksjomatem dla tej metody i w pewnych przypadkach na pewno nie jest konieczne, jednakże w rozpatrywanych tu problemach jest wykorzystywane, gdyż znacząco upraszcza proces przekształcania grafu. W przypadku rezygnacji z tego założenia różnica między początkową strukturą grafu połączeń, a końcową może być niewielka. Graf, w którym będą dodatkowe połączenia pomiędzy węzłami typu shell, należącymi do różnych hubów, można potraktować jako posiadający dodatkowe huby[1]. Dlatego też przy stosowaniu tego typu nierozłącznych struktur należałoby narzucić silne ograniczenia na liczbę wspólnych wierzchołków, co z kolei prowadzi do powstawania najczęściej sztucznych ograniczeń niewynikających bezpośrednio ze struktury problemu (chyba, że właśnie takie uwarunkowania w rozpatrywanym problemie występują) i jego komplikacji. Z tego też powodu tego typu rozwiązania nie będą rozpatrywane w tej pracy, jednakże zadania, w których podgrafy typu shell nie są rozłączne są przedmiotem dalszych badań autorów. Wydaje się również, że w praktycznych rozwiązaniach często potrzebne będą pewne modyfikacje otrzymanych rozwiązań, przystosowujące je do lokalnych wymagań, które będą wymuszały zastosowanie dodatkowych połączeń do innych hubów lub dodatkowych połączeń między węzłami lokalnymi, ewentualnie wprowadzenia dodatkowych hubów. Sytuacja taka może zaistnieć np. w przypadkach, gdy na podstawie istniejących połączeń nie da się zrealizować założonej struktury, bo graf połączeń jest zbyt „rzadki” lub też będą konieczne modyfikacje w trakcie działania struktury kernel and shell na skutek zmiany struktury przewozów w danym regionie.

W opisanych w tym rozdziale wariantach metody omówione są różne jej realizacje.

3.1 Graf nieskierowany i nieważony

3.1.1 Metoda hub and spoke

Struktura połączeń „*hub and spoke*” jest szczególnym przypadkiem struktury „*kernel and shell*”. Struktura ta jest najczęściej znacznym uproszczeniem w stosun-

[1] Oczywiście można również rozpatrzyć strukturę z różnymi kategoriami węzłów typu kernel i wyróżnić węzły typu kernel niższej kategorii.

ku do grafu wejściowego, gdyż za istotne przyjmuje się tylko połączenia między węzłami typu kernel, zwanymi tu hubami oraz lokalnymi połączeniami węzłów typu shell (spoke) do swojego huba.

Otrzymana struktura ma elementy, które są liśćmi dołączonymi do swoich hubów, albo hubami. Połączenie dwóch liści odbywa się przez co najmniej jeden hub w przypadku, gdy liście należą do tego samego pierścienia oraz przez przynajmniej dwa huby w przypadku, gdy liście należą do różnych pierścieni. Ilustruje to rys. 4.1. Liczba odwiedzanych po drodze hubów zależy w dużej mierze od liczby połączeń w tworzonym przez nie podgrafie i najmniejsza jest wtedy, gdy tworzą klikę (2), a największa, gdy jest to lista. Wtedy, w skrajnie niekorzystnym przypadku trzeba po kolei odwiedzić wszystkie huby. Takie rozwiązanie jest jednak najczęściej mało sensowne i użyteczne. Najprawdopodobniej transformowany graf jest bardzo rzadki i z pewnością konieczne jest rozważenie dodania dodatkowych połączeń między hubami lub poddawanie go transformacji nie ma sensu.

W przypadku grafu nieważonego istotny jest tylko fakt istnienia połączenia pomiędzy węzłami. Węzły są przydzielane do podgrafów tylko na podstawie tego faktu, nie jest brana pod uwagę jakakolwiek odległość, położenie, znaczenie połączenia. Z tego też powodu nie zawsze taka metoda może i powinna być stosowana. Jeśli istotne są jakieś inne czynniki poza faktem istnienia bądź braku połączenia, to lepsza będzie metoda opracowana dla grafu ważonego.

Metoda *hub and spoke* transformacji grafu połączeń jest przydatna w sytuacjach, gdy w grafie połączeń istotne są głównie połączenia hubów ze swoimi węzłami peryferyjnymi, transfery między węzłami peryferyjnymi są na tyle rzadkie lub nieobciążające huby, że nie są potrzebne dodatkowe połączenia między nimi. Wszelkie transfery lokalne przeprowadzane są przez lokalny hub. Metody transformacji do omawianej struktury można użyć na kilka sposobów:

- liczba hubów i ewentualnie kandydaci na huby lub ich część są wyznaczane wstępnie przed transformacją, pozostałe węzły wybiera algorytm;
- poszukiwana jest minimalna liczba hubów, która tworzy przynajmniej graf spójny z wszystkimi węzłami przydzielonymi do hubów (widoczne jest, że metoda pierwsza powinna operować liczbami większymi niż wartość otrzymana w tej metodzie);
- liczba hubów jest dobierana przez algorytm na podstawie struktury grafu wejściowego i innych parametrów (stopień połączenia hubów między sobą, licznościach węzłów w podgrafach typu spoke. itp.).

Powody wyboru którejś z powyższych metod transformacji są dość oczywiste. Metoda druga powinna być stosowana raczej jako metoda umożliwiająca oszacowanie dolnej liczby możliwych hubów, otrzymane przy jej pomocy rozwiązania są raczej mało praktyczne, ale stanowią podstawę dla innych metod.

Metoda pierwsza wraz z np. informacją o minimalnej możliwej liczbie hubów i ewentualnie pewnymi węzłami wyznaczonymi a priori do tej funkcji jest najbardziej użyteczna, gdyż taka sytuacja występuje najczęściej, np. chcąc poprawić funkcjonowanie pewnego istniejącego układu komunikacyjnego z pewnością dobrze jest wybrać na huby powstałe już centra przesiadkowe i dobudować brakujące niż na

siłę tworzyć wszystko od nowa. Z kolei metoda trzecia jest odpowiednia w sytuacji zupełnie nowo tworzonych struktur, którym chcemy narzucić pewne ogólne ograniczenia i stworzyć strukturę odpowiednia dla projektowanej sieci połączeń.

Pewne problemy powstają, gdy transformowany graf ma bardzo mało połączeń i trudno jest go przekształcić do odpowiedniej struktury, gdyż pojawiają się węzły niepodłączone do żadnego z wydzielonych hubów. W takiej sytuacji możliwych jest kilka rozwiązań. Niepodłączone węzły zostaną przydzielone do pewnych hubów na podstawie mniej istotnych kryteriów np. równoważenia liczby węzłów podłączonych do hubów, natomiast konieczne jest wtedy rozważenie utworzenia nowego połączenia między niepodłączonym węzłem, a hubem do którego został przydzielony. Innym możliwym rozwiązaniem jest utworzenie dodatkowego huba (z węzłów, które stały się liśćmi), do którego ten węzeł jest podłączony.

3.1.2 Metoda alfaklikowa

Metoda alfaklikowa jest drugą z rozważanych wersji struktury kernel and shell. Jako wersja struktury kernel and shell również posiada silnie połączony szkielet, budujący strukturę węzłów tranzytowych – kernel - oraz powiązane z każdym węzłem kernela struktury typu shell. Węzły tranzytowe stanowią silnie powiązany podgraf, każdy zaś z nich posiada swoje węzły lokalne. Podgrafy węzłów lokalnych nie stanowią jednak struktury liści powiązanych z centrum, a są podgrafami z powiązaniami lokalnymi i stanowią α-kliki o założonej wartości α. Praktycznie wartość α dla pewnego podgrafu G o symetrycznej, zero-jedynkowej ($a_{ij} \in \{0, 1\}$, przyjmujemy, że zawsze $a_{ii}=1$) macierzy sąsiedztwa A o rozmiarach $n \times n$ można policzyć przy użyciu następującego wzoru:

$$\alpha = \frac{1}{n} \min_{j} \left(\sum_{i=0}^{n-1} a_{ij} \right), \quad j = 0 \ldots n-1 \tag{3.1}.$$

Strukturę taką ilustruje rys. 4.2. Podobnie jak w metodzie *hub and spoke*, w grafie połączeń istotna jest tylko obecność połączenia, a nie jego waga.

Strukturę alfaklikową warto jest zastosować w sytuacji, gdy bardzo istotne są również połączenia między lokalnymi węzłami i nie można całością transferów lokalnych obciążyć lokalnego huba.

Zastosowana tu metoda opiera się na pojęciu alfakliki, opisanej pełniej w rozdziale 2.2. Metoda polega na podziale grafu na alfakliki o zadanej lub wyliczonej wartości α oraz wydzieleniu podgrafu kernela o jak największej liczbie połączeń pomiędzy tworzącymi go węzłami. W ten sposób, zależnie od wartości parametru α uzyskamy podział grafu wejściowego na różną liczbę podgrafów typu shell. Dodatkowo można podać uzupełniające warunki wymuszające np. równoważenie wielkości otrzymywanych struktur itp.

Metoda ta może być używana na dwa sposoby:

- część lub wszystkie węzły kernela oraz ich maksymalna liczba są ustalone a priori, algorytm wyszukuje kandydatów na brakujące węzły kernela oraz znajduje zawartość podgrafów typu shell;
- narzucana jest wartość α dla podgrafów typu shell, ich zawartość oraz podgraf węzłów typu kernel jest dobierana tak, aby spełnić narzucone ograniczenie i jednocześnie utworzyć jak najlepiej połączony podgraf typu kernel.

Pierwszy przypadek jest prawdopodobnie bardziej użyteczny niż drugi, gdyż w rzeczywistych sytuacjach często wiadomo jest, które węzły mają stać się elementami kernela (istniejące duże węzły komunikacyjne, duże i ważne miasta – stolice państw, prowincji, itp.), ale jeśli struktura połączeń nie jest wcześniej znana i graf połączeń jest gęsty, wtedy druga metoda może być łatwiejsza do zastosowania i późniejszego wykorzystania.

3.2. Graf ważony

3.2.1. Metoda "hub and spoke" w grafie ważonym

W ogólnych zarysach metoda ta jest zbliżona do wersji opisywanej dla grafu nieważonego. Główną różnicą jest tu uwzględnianie nie tylko samego faktu połączenia między węzłami, ale również jego wagi (zakładamy, że wagi są nieujemne). W wielu problemach ma to decydujące znaczenie, dlatego też poświęcono temu zagadnieniu osobny podrozdział.

W metodzie tej otrzymuje się podobną, gwiazdową strukturę sieci połączeń, jednakże duże znaczenie w kryterium podziału grafu na struktury typu *hub* lub *spoke* mają wagi połączeń między węzłami. W związku z tym o przydziale do odpowiedniego huba decyduje oczywiście obecność połączenia (jego brak uniemożliwia taki przydział), jednakże w przypadku istnienia w grafie początkowym połączeń do kilku hubów, decydujące będzie to, o największej wadze i tam też zostanie przydzielony węzeł. W efekcie powstaje struktura, w której lepiej można odwzorować istniejące uwarunkowania związane z odległością, natężeniem ruchu, przepływem danych, osób lub towarów niż przy użyciu metody opracowanej dla grafu nieważonego.

Metody transformacji do omawianej struktury można użyć na kilka sposobów, podobnie jak w przypadku grafu bez wag. Wydaje się, że poszukiwanie podziału o minimalnej liczby hubów w tej wersji nie ma większego sensu, gdyż będzie to podział taki sam, jak w przypadku metody nieuwzględniającej wag (uwzględnienie wag nie zmniejszy minimalnej liczby hubów możliwej do wydzielenia w grafie).Wobec tego wyróżniono dwa sposoby zastosowania metody:

- liczba hubów lub huby lub ich część są wyznaczane wstępnie przed transformacją, pozostałe węzły wybiera algorytm;
- liczba hubów jest dobierana przez algorytm na podstawie struktury grafu wejściowego i innych parametrów (stopień połączenia hubów między sobą, licznościach węzłów w podgrafów itp.).

Charakterystyka i przydatność obu sposobów zastosowania opisywanej metody jest analogiczna jak w przypadku metody bez wag opisanej w punkcie 3.1.1.

3.2.2. Metoda alfaklikowa w grafie ważonym

Metoda alfaklikowa w grafie ważonym jest również rozszerzeniem metody alfaklikowej bez uwzględniania wag połączeń, opisanej w punkcie 3.2. Możliwe jest zastosowanie jej w kilku wersjach. W wariancie najbardziej zbliżonym do metody w wersji nieważonej występuje taka sama metoda obliczania wartości α (wg wzoru 3.1), a jedynie promowane są wersje rozwiązania w których sumy wag połączeń są większe niż w innych możliwych. W drugiej wersji do obliczeń wykorzystuje się zmodyfikowany wzór na wartość współczynnika α_w (3.2) z oczywistym naciskiem na maksymalizację wszystkich wag połączeń w generowanej strukturze. α_w obliczone dla podgrafu G o kwadratowej macierzy wag A (będącej uogólnieniem macierzy sąsiedztwa) o wyrazach $a_{ij} \geq 0$ (przyjmujemy, że $a_{ii}=1$) i rozmiarze n, wyraża się wzorem:

$$\alpha_w = \frac{1}{2n} \min_j \left(\sum_{i=0}^{n-1} \frac{1}{a_{\max}} (a_{ij} + a_{ji}) \right), \quad j = 0...n-1 \qquad (3.2).$$

We wzorze tym macierz wag A określa istnienie ($a_{ij}>0$) i siłę połączenia między węzłami, otrzymywane wartości α_w są zaś znormalizowane przez podzielenie wartość maksymalnej wagi a_{max}, więc otrzymywana wartość α w dalszym ciągu zawarta jest w przedziale $(0, 1\rangle$. Możliwe są również inne sposoby normalizacji wartości parametru α_w.

Metoda ta może być, podobnie jak w wersji nieważonej, używana na dwa sposoby:

- węzły kernela lub ich maksymalna liczba są ustalone a priori, a algorytm wyszukuje kandydatów na ewentualne brakujące węzły kernela oraz znajduje zawartość podgrafów typu shell;
- narzucana jest wartość α/α_w dla podgrafów typu shell, ich zawartość oraz podgraf węzłów typu kernel jest dobierana tak, aby spełnić narzucone ograniczenie i jednocześnie utworzyć jak najlepiej (pod względem wag) połączony podgraf typu kernel.

Oba przypadki zastosowania opisywanej metody mają cechy analogiczne do metody bez wag opisanej w punkcie 3.1.2.

3.3 Metoda „kernel and shell" w grafie skierowanym

Mimo, że opisywane metody zostały opracowane do zastosowania w grafach nieskierowanych, to jednak można spróbować rozszerzyć je również na grafy skierowane. W grafie skierowanym możliwe jest wykorzystanie metody kernel and shell w obu opisywanych wariantach – metodzie alfaklikowej i metodzie *hub and spoke*. Pojawiają się, co prawda, trudności z interpretacją pewnych pojęć w sytuacjach, gdy np. istnieją tylko połączenia w jedną stronę lub połączenia ujemne. Można założyć, że połączenie tylko w jedną stronę jest połową połączenia w obie strony i aby policzyć np. wartość α w wersji skierowanej można zastosować wzór (3.3):

$$\alpha_s = \frac{1}{2n}\min_j\left(\sum_{i=0}^{n-1}\left(\left|a_{ij}\right|+\left|a_{ji}\right|\right)\right), \quad j=0...n-1 \tag{3.3}$$

$$\alpha_s = \frac{1}{2n}\min_j\left(\sum_{i=0}^{n-1}\frac{1}{a_{\max}}\left(\left|a_{ij}\right|+\left|a_{ji}\right|\right)\right), \quad j=0...n-1 \tag{3.4}$$

w wersji bez uwzględniania wag, lub wzór (3.4) w wersji ważonej. W przypadku różnych wag w obie strony wypadkowa wielkość będzie ich średnią arytmetyczną. Możliwe jest również zażądanie istnienia połączeń w obu kierunkach i uwzględniania tylko takich połączeń. Wartości ujemne nie stanowią problemu, gdyż określają kierunek transferów, wobec czego we wzorach należy użyć wartości bezwzględnych. W ten sposób możliwe jest użycie metody alfaklikowej zarówno w wersji z założoną wartością α jak i założoną liczbą węzłów typu kernel.

Trudniejsza sytuacja jest w przypadku metody *hub and spoke*, w której mogą wystąpić opisywane wyżej problemy z interpretacją połączeń tylko w jedną stronę. W metodzie *hub and spoke* ważne jest istnienie połączeń pomiędzy każdym węzłem typu spoke i jego hubem. W przypadku, gdy połączenie istnieje tylko w jedną stronę jest to pewien problem. Dlatego też należy założyć, który rodzaj połączenia jest odpowiedni dla rozpatrywanego problemu i wtedy możliwe jest stosowanie obu wersji metody hub and spoke tak, jak to opisano w 3.2.1.

4. Metody ewolucyjne rozwiązywania problemów typu kernel & shell

4.1. Metoda wykorzystująca strukturę α-klikową przetwarzania grafu połączeń do postaci kernel and shell w grafie nieważonym

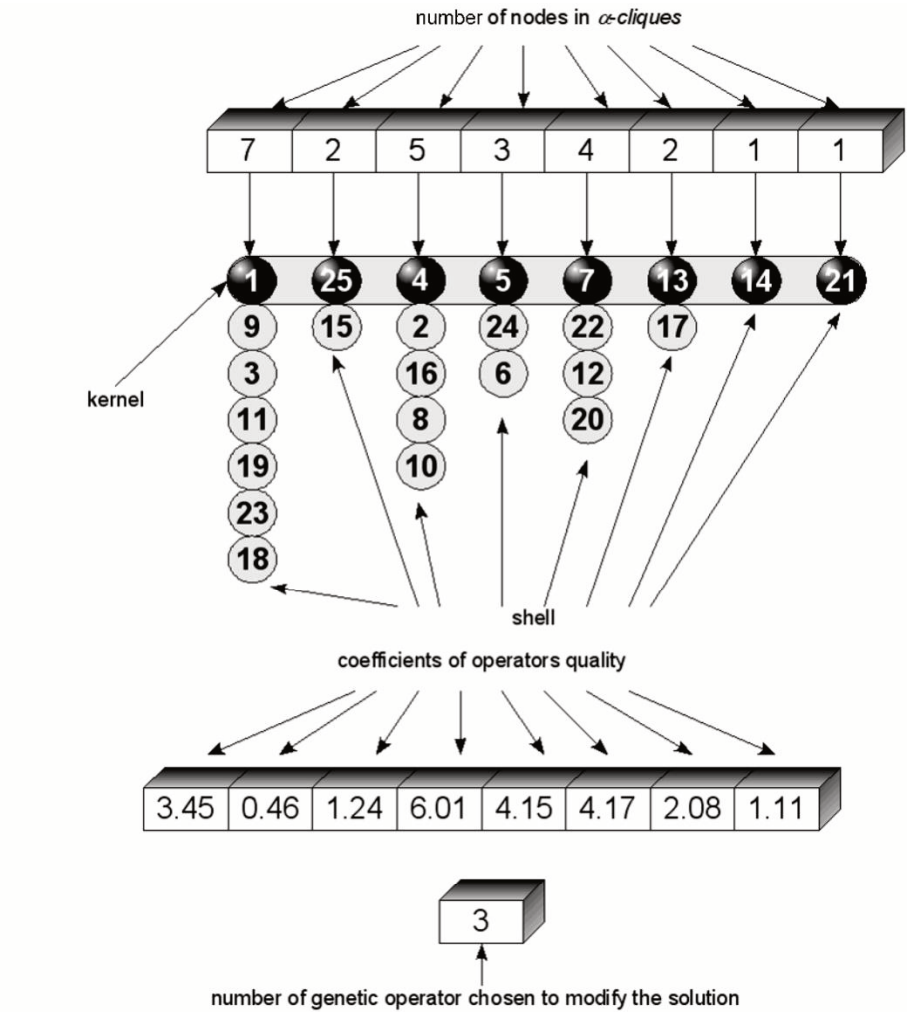
α-klikowa metoda przetwarzania grafu połączeń do struktury kernel and shell zawiera dwie możliwości takiej transformacji: wersję z ustaloną a priori liczbą hubów i metodę z założoną wartością α dla podgrafów składających się z liści. Tego typu strukturę połączeń warto jest stosować wtedy, gdy transfery pomiędzy węzłami podłączonymi do węzła typu hub są na tyle duże, że w sposób znaczący obciążałyby hub i wobec tego należy umożliwić niezależne od huba transfery pomiędzy węzłami.

4.1.1. Struktura α-klikowa z narzuconą liczbą lub narzuconymi węzłami typu kernel

Zakodowanie problemu, wykorzystywane w algorytmie ewolucyjnym, rozwiązującym ten problem przedstawione zostało na rys. 4.1. Pokazana tam struktura osobnika zawiera tablice z określona liczbą α-klik, tworzących shell oraz wydzielonymi (wybranymi a priori lub przez program) węzłami – reprezentantami w strukturze typu kernel.

Podgraf węzłów typu kernel tworzy strukturę o jak największej liczbie krawędzi między jego węzłami. Metoda stara się maksymalizować tę wielkość W przypadku narzucenia konkretnych węzłów, z których ma składać się struktura typu kernel, nie jest oczywiście możliwe powiększenie liczby połączeń pomiędzy nimi, na-

tomiast, jeśli narzucona jest tylko pewna ich liczba lub część kandydatów na huby, to metoda ma możliwość tak dobrać pozostałe wierzchołki, aby liczba tych połączeń była jak największa.



Rys. 4.1. Zakodowanie rozwiązania (osobnika) w metodzie opartej na strukturze α-klikowej z ustalonymi węzłami typu kernel.

W idealnym przypadku węzły typu kernel powinny tworzyć klikę, ale w rzeczywistych przypadkach, gdy graf połączeń jest dość rzadki, istotne jest, aby wierzchołki te tworzyły przynajmniej graf spójny. Warunek ten jest sprawdzany w trakcie obliczeń i wymuszany przy użyciu odpowiedniej funkcji kary (4.1). W symulacjach komputerowych użyta została następująca funkcja celu:

$$Q_1 = \alpha_{\min} - (1-\alpha_K) - \frac{1}{k}\sum_{i=1}^{n}(k_i - l_i)$$

$$\max Q = \begin{cases} \dfrac{Q_1}{m} & \text{gdy} \quad Q_1 < 0 \\ Q_1 * m & \text{gdy} \quad Q_1 \geq 0 \end{cases} \tag{4.1}$$

gdzie:
n – liczba narzuconych podgrafów typu shell,
k – liczba wszystkich węzłów w grafie,
k_i – liczba węzłów w i-tej α-klice,
l_i – liczba połączeń pomiędzy hubem i-tego shella a pozostałymi jego węzłami,
m – liczba spójnych składowych podgrafu kernela (powinna wynosić 1),
α_{min} – minimalna wartość α w wyznaczonych α-klikach tworzących shell,
α_K – wartość α wyznaczonej α-klice tworzącej kernel.

Dlaczego został wybrany taki wzór funkcji dopasowania, a nie inny? Tak, jak to już zostało wspomniane, wzór jest tworem sztucznym, który ma zapewnić odpowiednie parametry otrzymanego rozwiązania, nie wynika zaś jednoznacznie z jakichś właściwości optymalizowanego zadania. Taka postać wzoru jest też produktem pewnej ewolucji, w której metodą prób i błędów powstawała formuła, dzięki której powstawały rozwiązania o odpowiednich cechach. Oczywiście na pewno możliwe jest zastosowanie innych jego postaci z prawdopodobnie lepszym nawet skutkiem, szczególnie, jeśli użytkownik będzie miał nieco inne oczekiwania. W podobny sposób powstały również i pozostałe wzory na funkcje celu w tej pracy, poza przypadkiem prostej minimalizacji liczby możliwych do zastosowania hubów.

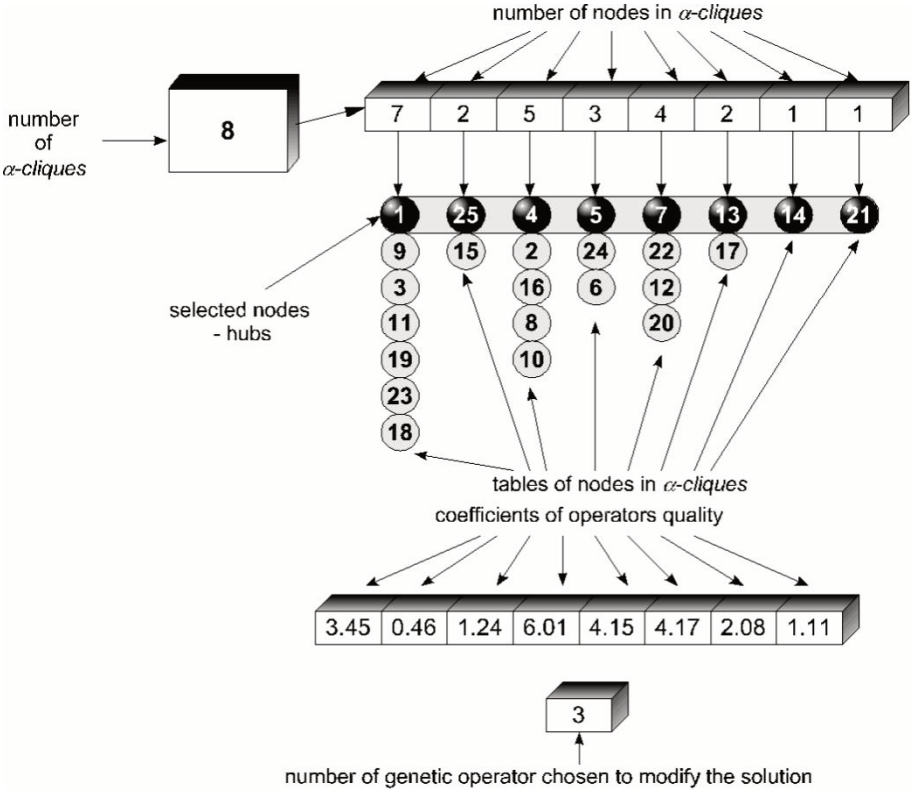
Specjalizowane kodowanie problemu w AE wymaga zastosowania specjalizowanych operatorów genetycznych, modyfikujących populację rozwiązań. Do rozwiązania omawianego problemu zaprojektowano następujące operatory genetyczne:

- mutacja – losowa wymiana losowo wybranych węzłów w wylosowanych podgrafach (α-klikach),
- przeniesienie losowo wybranego węzła pomiędzy wybranymi podgrafami
- wylosowanie nowego węzła typu kernel w wybranej α-klice (ten operator jest nieaktywny dla ustalonych a priori węzłów typu kernel),
- „inteligentne" przeniesienie – wykonywane tylko wtedy, gdy operacja przynosi poprawę funkcji dopasowania zadania.

4.1.2. Struktura α-klikowa z narzuconą wartością α

W przypadku tak postawionego zadania, brak jest narzuconej liczby lub samych węzłów typu kernel, natomiast narzucona jest minimalna wartość α na podgrafy tworzące shell. W związku z tym program sam dobiera strukturę rozwiązania, aby być w zgodzie z tym warunkiem. Liczba węzłów typu kernel, a co

za tym idzie, liczba struktur typu shell (α-klik) nie jest znana i może być zmienna w trakcie obliczeń, jednakże program, optymalizując funkcję dopasowania (4.2) z uwzględnieniem narzuconego ograniczenia na wartość α, sam tworzy odpowiednia strukturę grafu wynikowego. W związku z tymi zmianami struktura zakodowania tego problemu jest nieco inna niż w poprzednim przypadku, przedstawia ją rys. 4.2.



Rys. 4.2. Zakodowanie rozwiązania (osobnika) w metodzie opartej na strukturze α-klikowej z liczbą węzłów typu kernel określoną pośrednio przez podane parametry.

Funkcja celu użyta do rozwiązania tego zagadnienia:

$$\max Q = \frac{1}{n^* m}\sum_{i=1}^{n}\left(k_i - \left|\frac{k}{n} - k_i\right| + \frac{l_i}{k_i} - 1 + \frac{h_i}{n} - 1\right) \tag{4.3}$$

gdzie:
n – liczba wydzielonych przez metodę grafów typu shell,
k – liczba wszystkich węzłów w grafie,
k_i – liczba węzłów w i-tej α-klice,
l_i – liczba połączeń pomiędzy hubem i-tego shella a pozostałymi węzłami w tym shellu,
m – liczba spójnych składowych podgrafu kernela (powinna wynosić 1),

h_i – liczba połączeń pomiędzy hubem i a pozostałymi hubami.

Przedstawiona powyżej funkcja dopasowania (4.3) promuje rozwiązania o jak najmniejszej liczbie wydzielonych podgrafów typu shell (wartość funkcji dzielona jest przez n), najlepiej o liczbie wierzchołków jak najmniej odbiegającej od wartości średniej(k/n), zapewniając spójność podgrafu kernela (najlepiej jeśli m=1), maksymalizując liczbę połączeń pomiędzy węzłami typu kernel ($h_i/n-1$) oraz liczbę połączeń pomiędzy każdym węzłem typu kernel i węzłami w jego shellu ($l_i/n-1$).

Zastosowane operatory mogą w efekcie swojego działania tworzyć rozwiązania niedopuszczalne z α-klikami o niższej niż narzucona wartości α, w takiej sytuacji wprowadzone modyfikacje są anulowane i rozwiązanie pozostaje bez zmian. Stosowanie algorytmów naprawczych byłoby w tej sytuacji dość skomplikowane i kosztowne obliczeniowo, dlatego też zrezygnowano z ich stosowania. Rozwiązania przetwarzane przez algorytm są cały czas umiejscowione w dziedzinie rozwiązań dopuszczalnych i wobec tego zawsze istnieje pewność, że otrzymany wynik jest poprawny. Oczywiście zastosowanie możliwości występowania rozwiązań niedopuszczalnych i odpowiednich funkcji kary też jest możliwe, niemniej jednak nie zostało tu zastosowane. Użyto następujących operatorów genetycznych:

- mutacja – losowa wymiana losowo wybranych węzłów w wylosowanych podgrafach (α-klikach),
- przeniesienie losowo wybranego węzła pomiędzy wybranymi podgrafami;
- wylosowanie nowego węzła typu kernel w wybranej α-klice (ten operator jest nieaktywny dla ustalonych a priori węzłów typu kernel),
- konkatenacja – operator próbuje łączyć, szczególnie małe podgrafy, w jeden podgraf,
- „inteligentne" przeniesienie – wykonywane tylko wtedy, gdy operacja przynosi poprawę funkcji dopasowania zadania.

4.2. Struktura *hub and spoke* w grafie nieważonym

Zastosowania praktyczne struktury hub and spoke koncentrują się na przypadkach, w których transfery pomiędzy węzłami shella są nieistotne i nie obciążają istotnie huba, wobec tego bezpośrednie powiązania miedzy nimi są niepotrzebne. Wobec tego struktury shell w tym przypadku nie są α-klikami, lecz strukturami składającymi się z korzenia (hub) i liści (spokes). Korzenie, czyli huby są natomiast połączone ze sobą jak najgęściej i najlepiej, żeby tworzyły graf pełny, podobnie jak w przypadku struktury α-klikowej w kernel and shell.

4.2.1 Struktura hub and spoke z narzuconą liczbą hubów

Zgodnie z ogólną zasadą charakteryzującą opisywany sposób podejścia do tworzenia struktur kernel and shell w grafach połączeń, huby komunikacyjne powinny tworzyć sieć jak najgęściej połączonych węzłów, najlepiej klikę lub α-klikę o wysokiej wartości α, a w ostateczności przynajmniej podgraf spójny.

W rozpatrywanym przypadku nie tylko liczba, lecz również pewne węzły mogą być narzucone, np. istniejące już węzły kolejowe lub lotnicze, nawet jeśli według obiektywnych kryteriów powinny być zastąpione innymi, to jednak w praktyce koszty budowania nowych byłyby zbyt wielkie i z pewnością należy pozostawić i wykorzystać już istniejące.

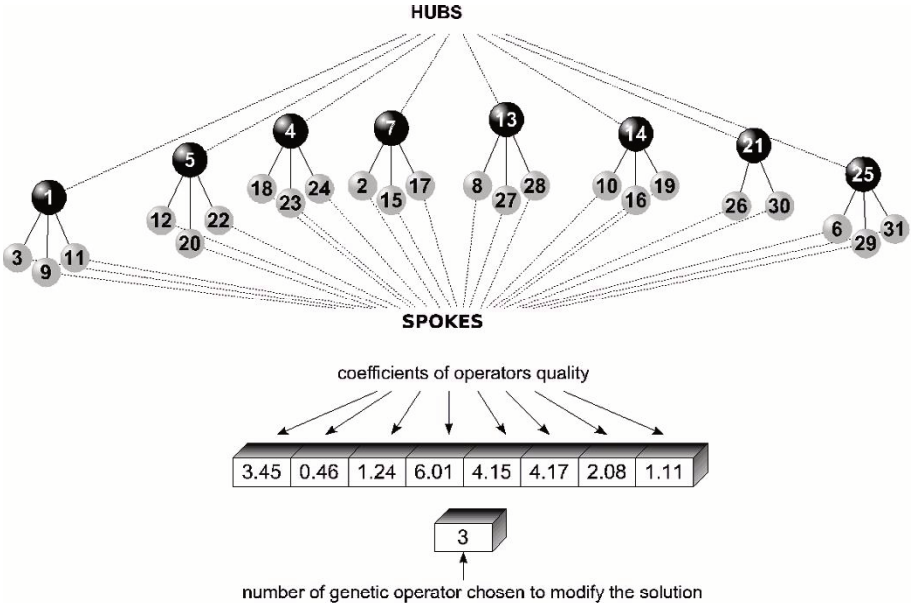
Węzły shella (*spokes*) są natomiast liśćmi podłączonymi do swoich hubów. W rozwiązaniu zadania istotne jest, aby wszystkie węzły typu spoke zostały powiązanie z hubami. W pewnych sytuacjach np., gdy graf połączeń jest bardzo rzadki i podana liczba hubów jest mniejsza od minimalnej, zapewniającej spójność grafu wynikowego lub gdy algorytm nie może znaleźć rozwiązania dopuszczalnego[2], to może się zdarzyć, że pozostaną pewne węzły niepowiązane z istniejącymi hubami, wtedy trzeba rozpatrzyć możliwość dodania dodatkowych hubów lub dodatkowych połączeń tak, aby powstałe rozwiązanie tworzyło graf spójny. Reakcję programu na pojawiające się rozwiązania niedopuszczalne można zdefiniować dwojako. W pierwszym przypadku program będzie starał się minimalizować liczbę niepodłączonych węzłów typu spoke, co można łatwo zapewnić przez dodanie do funkcji dopasowania członu, będącego funkcją kary zależną od liczby niepodłączonych węzłów. W drugim przypadku można pozwolić programowi wybrać dodatkowe huby, dzięki którym nie pojawią się niepodłączone węzły typu spoke. Strukturę zakodowania tego problemu w algorytmie ewolucyjnym przedstawia rys. 4.3. Rozwiązanie zawiera wybrane przez użytkownika lub program huby o określonej liczbie z dobranymi do nich przez program węzłami typu spoke. Oprócz tego osobnik zawiera dodatkowe dane, wymagane przez AG.

Funkcja dopasowania, zastosowana w tym przypadku metody *hub and spoke* promuje rozwiązania, w których węzły typu kernel (huby) tworzą między sobą jak najgęstszą sieć połączeń ($(h_i+1)/n$) z podgrafami typu shell o jak największych i wyrównanych rozmiarach ($|(k-n)/n-k_i|$) oraz o spójnym podgrafie węzłów typu kernel ($1/m$):

$$\max Q = \frac{1}{m}\sum_{i=1}^{n}\left(k_i - \left|\frac{k-n}{n} - k_i\right| + \frac{h_i+1}{n}\right) \tag{4.4}$$

gdzie:
n – ustalona liczba węzłów typu kernel,
m – liczba spójnych składowych powstałego grafu typu hub and spoke,
k_i – liczba węzłów należących do shella połączonego z hubem i,
k – liczba węzłów w całym grafie
h_i – liczba połączeń pomiędzy hubem i a pozostałymi hubami.

[2] Zawsze należy pamiętać, że opisywane metody rozwiązań są tylko metodami przybliżonymi, nie dającymi gwarancji znalezienia optimum lub rozwiązania dopuszczalnego.



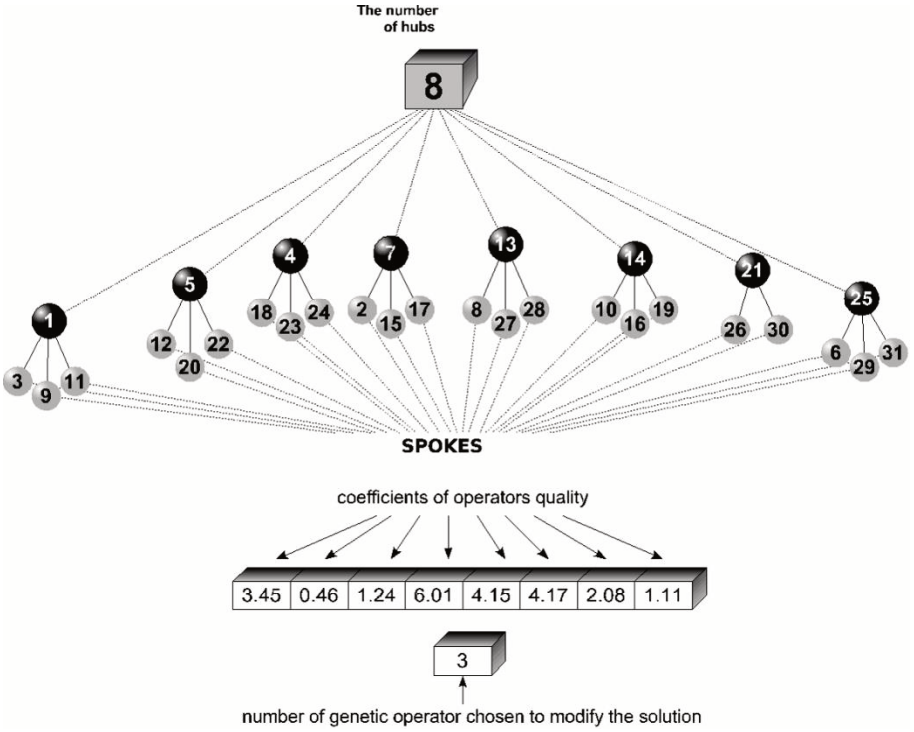
Rys. 4.3. Struktura rozwiązania zakodowanego w osobniku populacji AE dla przypadku metody hub and spoke z narzuconą liczbą hubów.

Opisywane zagadnienie można rozwiązać za pomocą operatorów genetycznych zbliżonych do zastosowanych w metodach α-klikowych, jednakże przy sprawdzaniu wykonalności operatora oraz prawidłowości rozwiązania po modyfikacji, sprawdzane są inne warunki. Przede wszystkim istotne jest, aby węzeł przenoszony pomiędzy różnymi podgrafami typu shell, miał połączenie z nowym hubem. Jeśli nie ma, to operacja jest anulowana, w przypadku stosowania operatora wielokrotnego, próba jest ponawiana kilkakrotnie z różnymi wylosowanymi węzłami przenoszonymi i miejscami docelowymi. Zestaw zastosowanych operatorów genetycznych składa się z:

- mutacji – wymiana losowo wybranych węzłów w wylosowanych podgrafach typu shell,
- przeniesienia losowo wybranego węzła pomiędzy wybranymi podgrafami typu shell,
- wymiany losowo wybranego huba na wylosowany węzeł typu spoke – ten operator jest nieaktywny, gdy węzły są wyznaczone przez użytkownika,
- "inteligentnego" przeniesienia – wykonywanego pomiędzy węzłami należącymi do różnych podgrafów typu shell tylko wtedy, gdy operacja przynosi poprawę funkcji dopasowania zadania.
- występują także wielokrotne wersje operatorów o losowanej z pewnego przedziału wartości liczbie powtórzeń.

4.2.2 Struktura typu hub and spoke z dobranymi przez algorytm węzłami typu kernel

W przypadku, gdy zezwolimy, aby metoda sama dobrała najlepszy układ węzłów typu hub i podłączonych do nich węzłów typu spoke, zakodowanie rozwiązania przybiera postać przedstawioną na rys. 4.4. Ponieważ liczba hubów nie jest znana, musi być ona dobrana przez algorytm i pamiętana w każdym rozwiązaniu, gdyż w każdym może być inna. Funkcję dopasowania dla tego przypadku przedstawia wzór (4.5).



Rys. 4.4. Struktura zakodowania rozwiązania dla przypadku z dobranymi przez algorytm węzłami typu kernel.

$$\max Q = \frac{1}{n * m} \sum_{i=1}^{n} \left(k_i - \left| \frac{k-n}{n} - k_i \right| + \frac{h_i + 1}{n} \right) \tag{4.5}$$

gdzie:
n – wyznaczona przez program liczba hubów,
m – liczba spójnych składowych grafu połączeń,
k_i – liczność shella i (węzłów przynależących do huba i),
k – liczba węzłów w całym grafie,
h_i – liczba połączeń pomiędzy hubem i a pozostałymi hubami.

Funkcja dopasowania (4.5) promuje rozwiązania o dużych rozmiarach

podgrafów typu shell, najlepiej o rozmiarze równym średniej liczbie węzłów w shellu ($|(k-n)/n-k_i|$), zapewniając spójność grafu (*1/m* – powinno być równe 1), maksymalizując liczbę połączeń pomiędzy hubami ((h_i+1)/*n*-1).

Zbiór operatorów genetycznych, stosowanych w tym przypadku zawiera:

- mutację – wymianę losowo wybranych węzłów w wylosowanych podgrafach typu shell,
- przeniesienie losowo wybranego węzła pomiędzy wybranymi podgrafami typu shell,
- wymiany losowo wybranego huba na wylosowany węzeł typu spoke,
- "inteligentne" przeniesienie – wykonywane pomiędzy węzłami należącymi do różnych podgrafów typu shell tylko wtedy, gdy operacja przynosi poprawę funkcji dopasowania zadania,
- konkatenacja – łączenie grup typu shell wraz z hubami w jedną całość,
- występują także wielokrotne wersje operatorów o losowanej z pewnego przedziału wartości liczbie powtórzeń.

4.2.3 Struktua typu hub and spoke z minimalną liczbą węzłów typu hub

Struktura *hub and spoke* z minimalna liczba węzłów jest szczególnym przypadkiem wersji z dobieranymi przez algorytm węzłami typu kernel. Zakodowanie problemu jest identyczne jak w p. 4.2.2, przedstawione na Rys. 4.4., natomiast funkcja dopasowania jest inna, przedstawia ją wzór 4.6:

$$\min Q = n * m \tag{4.6}$$

gdzie:
n – liczba wyznaczonych hubów,
m – liczba spójnych składowych grafu.

Funkcja dopasowania promuje w tym przypadku rozwiązania, które są grafami spójnymi i jednocześnie posiadają jak najmniej hubów. Algorytm stara się wystartować od rozwiązań dopuszczalnych, czyli dobiera tyle hubów, aby graf był spójny i minimalizuje ich liczbę, zachowując spójność. Jeśli nie da się utworzyć grafu spójnego na żadnym etapie obliczeń, to oczywiście otrzymane rozwiązanie nie jest konstruktywne i nie wnosi użytecznej informacji. Być może świadczy o niespójności całego grafu, co oczywiście można sprawdzić przed obliczeniami.

Zbiór operatorów genetycznych, stosowanych w tym przypadku jest identyczny jak w opisanym w punkcie 4.2.2.

4.3. Metoda wykorzystująca strukturę α-klikową przetwarzania grafu połączeń do postaci kernel and shell w grafie ważonym

W wielu przypadkach operowanie na grafach, w których wagi połączeń pomiędzy węzłami są nieistotne, może być sztuczne i słabo powiązane z rzeczywistością. Prawie zawsze odległość, przepustowość, obciążenie, niezawodność i tym podobne czynniki mają znaczenie przy transporcie różnego rodzaju dóbr, towarów, pasażerów, danych, a odbiciem tych zjawisk są analizowane

i transformowane grafy połączeń. Oczywiście podstawowe znaczenie w transformacji grafów mają ograniczenia, które wymuszają określoną postać i strukturę takiego grafu oraz spełnienie ograniczeń na spójność grafu, jednakże w wielu wypadkach możliwe są różne alternatywne konfiguracje grafu wynikowego i w tym przypadku uwzględnienie wag połączeń może decydować o tym, która z nich jest korzystniejsza, a która nie. W większości przypadków metody zastosowane w AE rozwiązującym zadania związane z grafami ważonymi są dość podobne do tych, omawianych przy okazji grafów nieważonych, dlatego też w następnych podrozdziałach omówione zostaną głównie różnice pomiędzy zastosowanymi rozwiązaniami, a tym, co opisano w odpowiednich podrozdziałach poświęconych grafom nieważonym.

4.3.1. Struktura α-klikowa z określoną a priori liczbą lub węzłami typu kernel w grafie ważonym

Podobnie jak w przypadku analogicznej struktury w grafie nieważonym, w grafie połączeń huby komunikacyjne powinny tworzyć sieć jak najgęściej połączonych węzłów, najlepiej strukturę przypominającą klikę lub α-klikę o wysokiej wartości α_w, a w ostateczności przynajmniej podgraf spójny, zapewniając spójność całemu grafowi. Jednakże w przypadku grafu ważonego zamiast α zdefiniowanego wzorem (3.1), użyta jest wartość α_w, w której nie liczy się tylko sam fakt połączenia pomiędzy węzłami, ale też jego waga (3.2).

Lokalizacja części lub wszystkich węzłów typu kernel jest narzucona, zgodnie zapewne z już istniejącymi obiektami. Jednakże elementy nienarzucone są wybierane tak, aby wagi połączeń pomiędzy nimi, a wybranymi uprzednio węzłami typu kernel były jak największe.

Węzły shella tworzą podgrafy - α-kliki o również jak najwyższej wartości parametru α, lecz również o przypisaniu do alfakliki decydują wagi pomiędzy węzłami. Preferowane są połączenia o większych wartościach wag. Rozwiązanie zawiera wybrane przez użytkownika lub program węzły typu kernel o określonej liczbie, z dobranymi do nich przez program węzłami shella. Oprócz tego osobnik zawiera dodatkowe dane, wymagane przez algorytmy ewolucyjne.

W symulacjach komputerowych użyta została następująca funkcja celu:

$$Q_1 = \alpha_{w\_\min} - (1 - \alpha_{w\_K}) - \frac{1}{k}\sum_{i=1}^{n}(k_i - l_i)$$

$$\max Q = \begin{cases} \dfrac{Q_1}{m} & \text{gdy} \quad Q_1 < 0 \\ Q_1 * m & \text{gdy} \quad Q_1 \geq 0 \end{cases} \tag{4.7}$$

gdzie:
n – liczba narzuconych grafów typu shell,
k – liczba wszystkich węzłów w grafie,
k_i – liczba węzłów w i-tej α-klice,

l_i – liczba połączeń pomiędzy hubem i-tego shella a pozostałymi węzłami w tym shellu,
m – liczba spójnych składowych podgrafu kernela (powinna wynosić 1),
α_{w_min} – minimalna wartość α ważonego w wyznaczonych α-klikach, tworzących shell,
α_{w_K} – wartość α ważonego w wyznaczonej α-klice, tworzącej kernel.

Funkcja celu (dopasowania) składa się z kilku elementów, które mają za zadanie zapewnić rozwiązania najbardziej odpowiadające warunkom zadania. W związku z tym funkcja dopasowania (4.7) stara się wypromować rozwiązania, w których podgraf węzłów, tworzących strukturę kernel ma wartość jak najbliższą 1 (maksymalizacja α_{w_K}). Oczywiście w przypadku rzadkiego grafu połączeń jest to niemożliwe do osiągnięcia, więc istotne jest, aby podgraf ten był przynajmniej spójny (z m=1). Jeśli znalezienie takiego rozwiązania również będzie niemożliwe, to może to oznaczać, że przetwarzany graf połączeń jest niespójny, choć oczywiście jest możliwe, że zastosowany algorytm genetyczny nie może znaleźć właściwego rozwiązania, gdyż jest to tylko metoda przybliżona. α-kliki tworzące shell również powinny charakteryzować się jak najwyższą wartością α_ω (funkcja celu maksymalizuje α_{w_min} – najniższą wartość α_w w wyznaczonych α-klikach shella) oraz jak najwięcej węzłów shell powinno mieć bezpośrednie połączenie ze swoim hubem (optymalna jest zerowa wartość sumy w ostatnim członie górnego wzoru (4.7)).

Specjalizowane kodowanie problemu w AE wymaga zastosowania specjalizowanych operatorów genetycznych, modyfikujących populację rozwiązań. Do rozwiązania omawianego problemu zaprojektowano następujące operatory genetyczne:

- mutacja – losowa wymiana losowo wybranych węzłów w wylosowanych podgrafach (α-klikach),
- przeniesienie losowo wybranego węzła pomiędzy wybranymi podgrafami;
- wylosowanie nowego węzła typu kernel w wybranej α-klice (ten operator jest nieaktywny dla ustalonych a priori węzłów typu kernel),
- "inteligentne" przeniesienie – wykonywane tylko wtedy, gdy operacja przynosi poprawę funkcji dopasowania zadania.

4.3.2. Struktura α-klikowa z narzuconą wartością α w grafie ważonym

W tym przypadku narzucona jest minimalna wartość α_w na podgrafy tworzące shell. Możliwe jest też narzucenie wartości α_w na węzły podgrafu typu kernel, lecz najprawdopodobniej łączne wmuszanie tych warunków nie powiedzie się. Liczba węzłów typu kernel, a co za tym idzie, liczba struktur typu shell (α-klik) nie jest znana i jest zmienna w trakcie obliczeń, jednakże program, optymalizując funkcję dopasowania (4.8) z uwzględnieniem narzuconego ograniczenia na wartość α_w, sam tworzy odpowiednia strukturę grafu wynikowego.

Funkcja celu użyta do rozwiązania tego zagadnienia:

$$\max Q = \frac{1}{n*m}\sum_{i=1}^{n}\left(k_i - \left|\frac{k}{n} - k_i\right| + \frac{l_i}{k_i} - 1 + \frac{h_i}{n} - 1\right) \tag{4.8}$$

gdzie:
n – liczba wydzielonych przez metodę grafów typu shell,
k – liczba wszystkich węzłów w grafie,
k_i – liczba węzłów w i-tej α-klice,
l_i – liczba połączeń pomiędzy hubem i-tego shella a pozostałymi węzłami w tym shellu,
m – liczba spójnych składowych podgrafu kernela (powinna wynosić 1),
h_i – liczba połączeń pomiędzy hubem i a pozostałymi hubami.

Przedstawiona powyżej funkcja dopasowania (4.8) promuje rozwiązania o jak najmniejszej liczbie wydzielonych podgrafów typu shell (wartość funkcji dzielona jest przez n), najlepiej o liczbie wierzchołków jak najmniej odbiegającej od wartości średniej(k/n), zapewniając spójność podgrafu kernela (najlepiej jeśli m=1), maksymalizując liczbę połączeń pomiędzy węzłami typu kernel ($h_i/n-1$) oraz liczbę połączeń pomiędzy każdym węzłem typu kernel i węzłami w jego shellu ($l_i/n-1$).

Zastosowane operatory mogą w efekcie swojego działania tworzyć rozwiązania niedopuszczalne z α-klikami o niższej niż narzucona wartości α, w takiej sytuacji wprowadzone modyfikacje są anulowane i rozwiązanie pozostaje bez zmian.

Stosowanie algorytmów naprawczych byłoby w tej sytuacji dość skomplikowane i kosztowne obliczeniowo, dlatego też zrezygnowano z ich stosowania. Rozwiązania przetwarzane przez algorytm są cały czas umiejscowione w dziedzinie rozwiązań dopuszczalnych i wobec tego zawsze istnieje pewność, że otrzymany wynik jest poprawny. Oczywiście zastosowanie możliwości występowania rozwiązań niedopuszczalnych i odpowiednich funkcji kary też jest możliwe, niemniej jednak nie zostało tu zastosowane. Użyto następujących operatorów genetycznych:

- mutacja – losowa wymiana losowo wybranych węzłów w wylosowanych podgrafach (α-klikach),
- przeniesienie losowo wybranego węzła pomiędzy wybranymi podgrafami;
- wylosowanie nowego węzła typu kernel w wybranej α-klice (ten operator jest nieaktywny dla ustalonych a priori węzłów typu kernel),
- konkatenacja – operator próbuje łączyć, szczególnie małe podgrafy, w jeden podgraf,
- „inteligentne" przeniesienie – wykonywane tylko wtedy, gdy operacja przynosi poprawę funkcji dopasowania zadania.

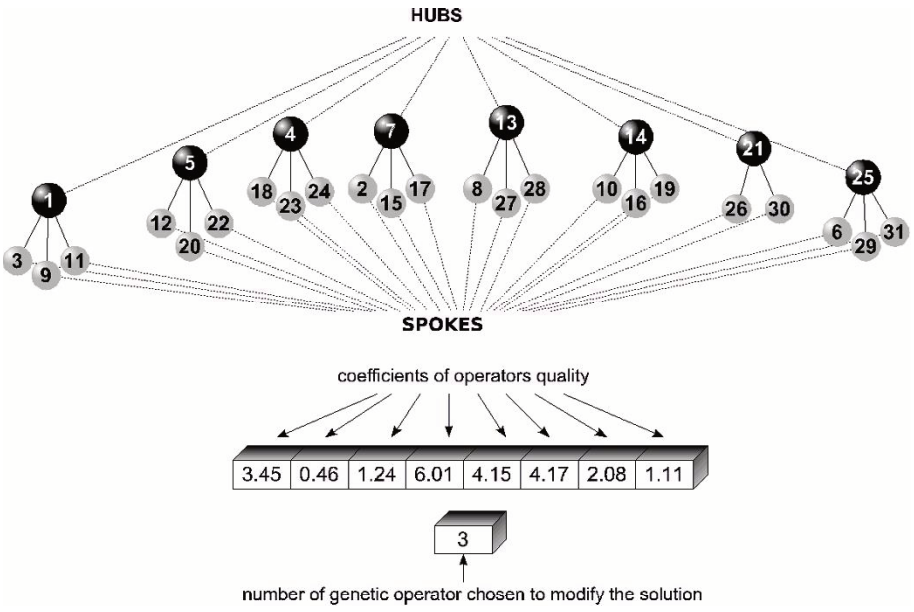
4.4. Struktura *hub and spoke* w grafie ważonym

Zastosowania praktyczne struktury hub and spoke w grafie ważonym nie odbiegają zasadniczo od wersji dla grafu nieważonego. Modyfikacja polega na tym,

że duże znaczenie w tej sytuacji mają wagi połączeń, które mają głos decydujący o przynależności konkretnego węzła do konkretnego huba. Sytuacje, w których wagi połączeń są istotne występują dość często, a z pewnością są częstsze niż te, gdy nie mają one znaczenia, wobec czego konieczne jest wyróżnienie takiego przypadku i przebadanie jego właściwości.

4.4.1 Struktura hub and spoke z określoną a priori liczbą lub konkretnymi węzłami kandydatami na huby

Podobnie jak w przypadku nieuwzględniającym wag, wymagane jest znalezienie struktury o silnie połączonym jądrze (kernel) węzłów typu hub (kernel) i dołączonych do nich jako liście węzłów peryferyjnych typu spoke (shell). Jednakże decydujący głos w sprawie wykorzystania konkretnego połączenia powinna mieć waga takiego połączenia. Nie wnikając już w to, co konkretnie ma ta waga oznaczać, przyjmujemy, że im większa jest jej wartość, tym lepiej (gdyby w jakimś zagadnieniu występował przeciwny przypadek, to zawsze można to zamienić, analogiczne jak problem minimalizacji można zmienić na problem maksymalizacji). Bardzo często jest też tak, że istniejące już ważne węzły komunikacyjne są kandydatami na sieć hubów. Wobec czego opisywany przypadek może mieć znaczenie wtedy, gdy chcemy przetworzyć strukturę połączeń w istniejącej już sieci komunikacyjnej, bez tworzenia nowych połączeń. Jeśli natomiast określona jest tylko liczba węzłów typu hub, to metoda sama wskaże najodpowiedniejszych jej zdaniem kandydatów.



Rys. 4.5. Struktura rozwiązania zakodowanego w osobniku populacji AE dla przypadku metody hub and spoke w grafie ważonym z narzuconą liczbą hubów.

Funkcja dopasowania (4.9) promuje rozwiązania o węzłach typu spoke jak najsilniej powiązanych ze swoimi hubami oraz o hubach jak najsilniej połączonych między sobą (odpowiednie sumy wag w), jednocześnie stara się zrównoważyć wielkość powstałych podgrafów typu spoke, pomniejszając wartość funkcji celu o odchyłkę wielkości podgrafów typu shell od wartości średniej. Podzielenie funkcji celu przez m ma zapewnić spójność grafu hubów.

$$\max Q = \frac{1}{m}\sum_{i=1}^{n}\left(\sum_{j=1}^{k_i}\left(w_{ij}\right)+\sum_{l=1}^{n}\left(w_{il}\right)-\left|k_{sr}-k_i\right|\right) \qquad (4.9)$$

$$k_{sr} = \frac{k-n}{n}$$

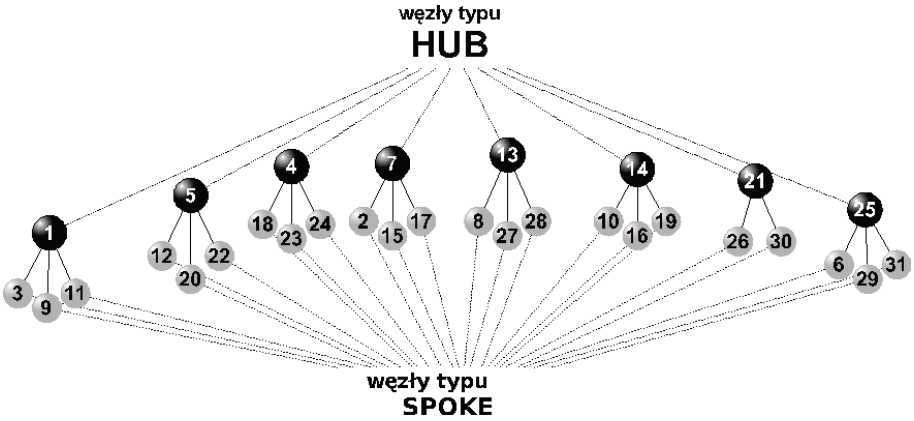
gdzie:
n – ustalona liczba węzłów typu kernel,
m – liczba spójnych składowych powstałego grafu typu hub and spoke,
k_i – liczba węzłów należących do shella połączonego z hubem i,
k – liczba węzłów w całym grafie,
k_{sr} – średnia liczba węzłów występująca w podgrafach typu shell ,

Zestaw zastosowanych operatorów genetycznych składa się z:

- mutacja – wymiana losowo wybranych węzłów w wylosowanych podgrafach typu shell,
- przeniesienie losowo wybranego węzła pomiędzy wybranymi podgrafami typu shell,
- wymiana losowo wybranego huba na wylosowany węzeł typu spoke – ten operator jest nieaktywny, gdy węzły są wyznaczone przez użytkownika,
- „inteligentne" przeniesienie – wykonywane pomiędzy węzłami należącymi do różnych podgrafów typu shell tylko wtedy, gdy operacja przynosi poprawę funkcji dopasowania zadania.
- występują także wielokrotne wersje operatorów o losowanej z pewnego przedziału wartości (zależnej od parametrów rozwiązania) liczbie powtórzeń.

4.4.2 Struktura typu hub and spoke z dobranymi przez algorytm węzłami typu kernel

W opisywanej tu wersji przekształcania grafu połączeń, metoda sama dobiera najlepszy układ węzłów typu hub i podłączonych do nich węzłów typu spoke. Zakodowanie rozwiązania w AE przybiera postać przedstawioną na rys. 4.6.



Rys. 4.6. Struktura zakodowania rozwiązania dla przypadku z dobranymi przez algorytm węzłami typu hub.

Strukturę powstającego grafu określa w pełni funkcja dopasowania. W trakcie obliczeń ewolucyjnych promuje osobniki o pożądanych cechach (w tym wypadku o najlepszych połączeniach), w efekcie czego powstaje graf o wymaganej strukturze. Oczywiście zawsze w algorytmie ewolucyjnym gra rolę przypadek i przy różnych symulacjach nawet dla tych samych danych powstają najczęściej różne rozwiązania, jednakże o podobnych własnościach. Funkcję dopasowania dla tego przypadku przedstawia wzór (4.10),

$$\max Q = \frac{1}{n_t * m} \sum_{i=1}^{n_t} \left(\sum_{j=1}^{k_i} (w_{ij}) + \sum_{l=1}^{n_t} (w_{il}) - \left| \frac{k - n_t}{n_t} - k_i \right| \right) \tag{4.10}$$

gdzie:
n_t – wyznaczana przez program liczba hubów, wartość zmienna w trakcie obliczeń (stąd indeks t),
m – liczba spójnych składowych grafu połączeń,
k_i – liczba węzłów typu spoke przynależących do huba i,
k – liczba węzłów w całym grafie,
w_{ij}, w_{il} – wagi połączeń odpowiednio: pomiędzy hubem i, a jego węzłem typu spoke o numerze j oraz wagi połączeń pomiędzy hubami y podgrafu typu kernel.

Funkcja dopasowania (4.10) promuje rozwiązania o jak najlepiej powiązanych węzłach typu spoke ze swoimi hubami oraz o hubach jak najsilniej połączonych między sobą (odpowiednie sumy wag w), jednocześnie stara się zrównoważyć wielkość powstałych podgrafów typu spoke ($|(k-n_t)/n_t-k_i|$), zapewniając spójność grafu ($1/m$ – powinno być równe 1) i jak najmniejszą liczbę samych hubów ($1/n_t$).

Oczywiście, taka postać funkcji celu nie wynika bezpośrednio z warunków zadania, a jest raczej wynikiem doświadczalnego jej doboru tak, aby otrzymywane

wyniki spełniały oczekiwania. Jest to bardzo ogólny przepis na wymaganą postać grafu, jednakże w połączeniu ze specjalizowanymi operatorami genetycznymi i selekcją rozwiązań, daje ciekawe rezultaty, przedstawione m.in. w rozdziale 5.

Zbiór operatorów genetycznych, stosowanych do rozwiązania zadania zawiera:

- mutację – wymianę losowo wybranych węzłów w wylosowanych podgrafach typu spoke.
- przeniesienie losowo wybranego węzła pomiędzy wybranymi podgrafami typu spoke.
- wymiany losowo wybranego huba na wylosowany węzeł typu spoke.
- „inteligentne" przeniesienie – wykonywane pomiędzy węzłami należącymi do różnych podgrafów typu spoke tylko wtedy, gdy operacja przynosi poprawę funkcji dopasowania zadania.
- konkatenacja – łączenie grup typu spoke wraz z hubami w jedną całość.
- występują także wielokrotne wersje operatorów, które polegają na kilkukrotnym powtórzeniu działania (losowanym z pewnego przedziału wartości) operatora dla tego samego rozwiązania w jednym pokoleniu.

5. Wyniki symulacji komputerowych

5.1 Przypadek grafu nieważonego

5.1.1. Dane wykorzystane do testów obliczeniowych

Jako dane testowe, przy braku dostępu do rzeczywistych danych ilustrujących przewozy lotnicze lub kolejowe, wykorzystaliśmy grafy testowe dostępne w BHOSLIB: Benchmarks with Hidden Optimum Solutions for Graph Problems (Maximum Clique, Maximum Independent Set, Minimum Vertex Cover and Vertex Coloring) – Hiding Exact Solutions in Random Graphs. Z dostępnych tam wielu przykładów wybraliśmy graf graf o 4000 wierzchołków i 7 425 226 krawędziach o Maximum Independent Set=100 i Minimum Vertex Cover=3900 – przykład o nazwie: (frb100-40.clq.gz).

Rozmiar wybranego grafu jest dość znaczny, a złożoność rozpatrywanego problemu zbliżona do takiej, jaką można napotkać przy rozpatrywaniu połączeń między większymi miastami w Europie lub na świecie.

5.1.2 Rezultaty otrzymane dla metody α-klikowej

5.1.2.1 Problem z pośrednio narzuconą strukturą podziału grafu

W pierwszym kroku zbadany został wpływ narzuconej wartości α na strukturę otrzymanego grafu wyjściowego. Uzyskane za pomocą algorytmu ewolucyjnego wyniki przedstawione są w Tabeli 1.

Tabela 1. Porównanie otrzymanych wyników

α		min	max	mediana	Liczba węzłów typu kernel	α_K	Liczba połączeń
Problem: frb100-40			Wierzchołki:4000	Krawędzie: 7 425 226			
0,75	LS	4000	4000	4000	1	1	7429226
	SKS	3727	3727	3727			
	SKK	1	1	1			
0,85	LS	4000	4000	4000	1	1	7429226
	SKS	3727	3727	3727			
	SKK	1	1	1			
0,95	LS	63	120	81	49	0,98	163136
	SKS	62	115	80			
	SKK	48	49	49			
1,00	LS	44	49	48	82	0,96	102869
	SKS	44	49	48			
	SKK	79	82	81			

Legenda: LS – liczność podgrafu typu shell, SKS – stopnie wierzchołków typu kernel w podgrafach typu shell, SKK - stopnie wierzchołków typu kernel w podgrafach typu kernel.

Niestety, dla tak dużych grafów niemożliwe jest narysowanie struktur (dokładniej: niemożliwe jest ich odpowiednio czytelne narysowanie) i wybór na tej podstawie najlepszych rozwiązań, podobnie jak prezentacja wyników w formie rysunków. W związku z tym w Tabeli 1 zamieszczone są pewne statystyki określające parametry otrzymanych grafów. Widoczne jest, że różne przyjęte wartości parametru α mają decydujący wpływ na otrzymane wyniki i w związku z tym sterowanie postacią otrzymanego grafu za pomocą tego parametru jest uzasadnione.

W dużym i gęstym grafie (4000 wierzchołków, problem frb100-40) znów widoczne jest, że nie ma sensu stosować wartości α mniejszych od 0,85, gdyż otrzymujemy tylko 1 węzeł typu kernel, czyli cały graf. Dla większych wartości α liczba węzłów typu kernel już szybko rośnie i osiąga liczbę 82 dla α=1,00. Maleją natomiast liczba wykorzystywanych połączeń z 7429226 do 102869 i nieznacznie wartość α_K z 1,00 do 0,96. W gęstym i dużym grafie zdecydowanie najtrudniej przeprowadzić obliczenia ewolucyjne, trwają one 3-4 dni.

5.1.2.2 Zagadnienie z narzuconymi parametrami podgrafu typu kernel

W tym przypadku w zadaniu narzucona jest liczba lub wyznaczone konkretne węzły typu kernel. Otrzymane wyniki przedstawia Tabela 2.

Tabela 2. Porównanie otrzymanych wyników

Liczba węzłów typu kernel		**min**	**max**	**Media-na**	α_{min}	α_K	**Liczba połączeń**
Problem: frb100-40		Wierzchołki: 4000		Krawędzie: 7 425 226			
5	LS	230	1871	537	0,90	1,00 / +	2319961
	SKS	211	1727	502			
	SKK	5	5	5			
20	LS	61	1275	72	0,90	0,95 / +	1165512
	SKS	61	1203	71			
	SKK	19	20	20			
100	LS	1	71	49	1,00	0,89	120893
	SKS	1	98	49			
	SKK	89	98	93			
200	LS	1	68	1	1,00	0,9	129864
	SKS	1	68	1			
	SKK	179	199	186			

Legenda: LS – liczność podgrafu typu shell, SKS – stopnie wierzchołków typu kernel w podgrafach typu shell, SKK - stopnie wierzchołków typu kernel w podgrafach typu kernel.

Podejście przeciwne do przedstawionego w poprzednim podpunkcie umożliwia prześledzenie właściwości transformacji grafu pod innym kątem.

We wszystkich badanych przypadkach widoczny jest spadek α_K ze wzrostem wielkości podgrafu typu kernel. Jest to zrozumiałe, gdyż dla większej liczby węzłów trudniej jest znaleźć silnie połączoną ich grupę. Odwrotnie zachowuje się α_{min}, jednakże tu zachodzi przeciwny proces, gdyż ze wzrostem wielkości podgrafu typu kernel, maleją wielkości lokalnych α-klik, dzięki czemu algorytm ewolucyjny łatwiej znajduje silniej powiązane grupy węzłów.

5.1.3 Wyniki otrzymane dla metody hub and spoke

5.1.3.1 Przypadek minimalnej liczby węzłów typu kernel/hub

Minimalna liczba węzłów typu kernel/hub, przy której można przeprowadzić transformację grafu połączeń, jest potrzebna raczej jako oszacowanie dolne możliwej ich liczby niż do celów praktycznych. Przez pojęcie „można przeprowadzić transformację" rozumiemy przeprowadzenie transformacji, po której otrzymuje się spójny podgraf typu kernel/hub z podłączonymi wszystkimi pozostałymi (jako węzłami typu shell) do swoich hubów. Otrzymane wyniki pokazuje Tabela 3. W przypadku problemu frb100-40 można mieć pewność, że

uzyskano wartość minimalną.

Tabela 3. Wyniki otrzymane przy poszukiwaniu minimalnego podziału grafu

Węzły typu kernel		**min**	**max**	**mediana**	**α_K**	**Liczba połączeń**
Problem: frb100-40	Wierzchołki: 4000			Krawędzie: 7 425 226		
2	LS	435	3563	435	1,00	7999
	SKS	435	3563	435		
	SKK	2	2	2		

Legenda: LS – liczność podgrafu typu shell, SKS – stopnie wierzchołków typu kernel w podgrafach typu shell, SKK - stopnie wierzchołków typu kernel w podgrafach typu kernel.

5.1.3.2 Przypadek z narzuconą liczbą węzłów typu kernel/hub

W tej metodzie otrzymuje się graf połączeń o strukturze hub and spoke z wyznaczoną a priori liczbą węzłów typu kernel/hub. Możliwe jest również wyznaczenie samych węzłów, lecz ten przypadek nie był rozpatrywany w przedstawionych w Tabeli 4 wynikach. Przedstawione w Tabeli 4 wyniki uwzględniają wartości minimalne, otrzymane w poprzednim podrozdziale, dlatego też wszystkie otrzymane grafy są spójne. We wszystkich przypadkach widoczny jest wzrost liczby połączeń w grafach po transformacji i raczej spadek wartości α_K wraz ze wzrostem liczby węzłów typu kernel/hub.

Tabela 4. Wyniki otrzymane dla przypadku z określoną a priori liczbą węzłów typu kernel/hub.

Liczba węzłów		**min**	**max**	**mediana**	**α_K**	**Liczba połączeń**
Problem: frb100-40	Węzły: 4000			Krawędzie: 7 425 226		
5	LS	799	799	799	1,00	8001
	SKS	799	799	799		
	SKK	5	5	5		
10	LS	399	399	399	0,90	8024
	SKS	399	399	399		
	SKK	9	10	10		
20	LS	199	199	199	0,90	8147
	SKS	199	199	199		
	SKK	18	20	20		
50	LS	79	79	79	1,00	9126
	SKS	79	79	79		
	SKK	50	50	50		
100	LS	39	39	39	0,99	12717
	SKS	39	39	39		
	SKK	99	100	99		
200	LS	19	19	19	0,97	27109
	SKS	19	19	19		
	SKK	194	200	196		

Legenda: LS – liczność podgrafu typu shell, SKS – stopnie wierzchołków typu kernel w podgrafach typu shell, SKK - stopnie wierzchołków typu kernel w podgrafach typu kernel.

5.1.3.3 Problem z dobraną przez algorytm liczbą węzłów typu kernel/hub

Tabela 5. Wyniki uzyskane dla metody, w której algorytm sam dobiera liczbę węzłów typu kernel/hub.

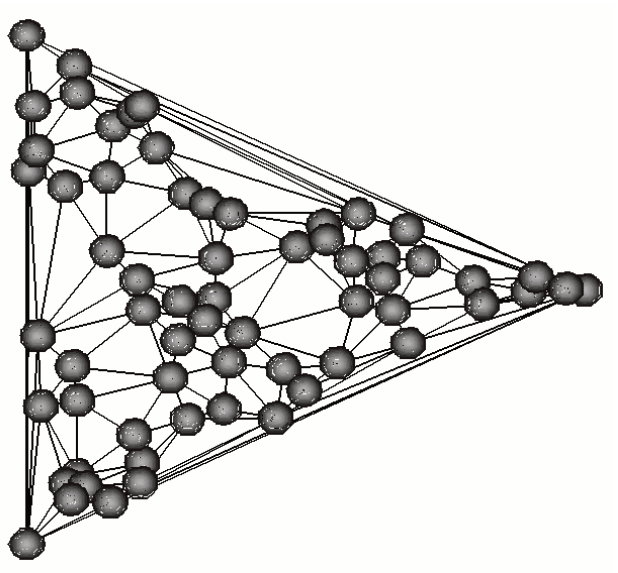
Uzyskana liczba węzłów typu kernel		**min**	**max**	**mediana**	α_K	**Liczba połączeń**
Problem: frb100-40		Węzły: 4000		Krawędzie: 7 425 226		
2	LS	1999	1999	1999	1,00	7999
	SKS	1999	1999	1999		
	SKK	2	2	2		

Legenda: LS – liczność podgrafu typu shell, SKS – stopnie wierzchołków typu kernel w podgrafach typu shell, SKK - stopnie wierzchołków typu kernel w podgrafach typu kernel.

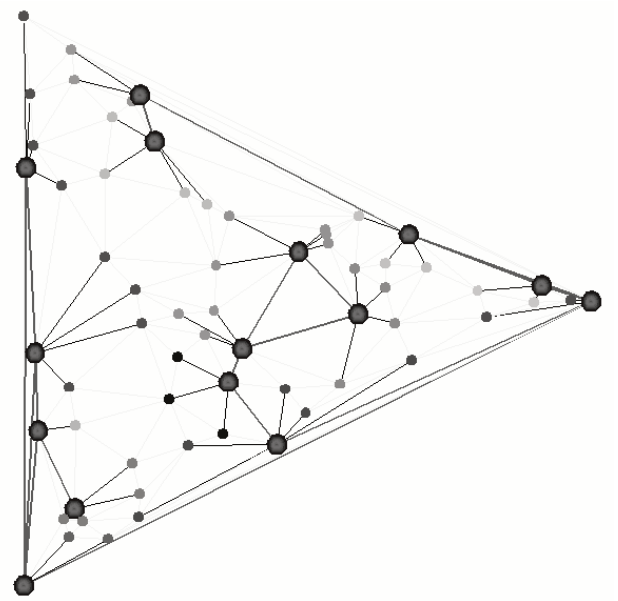
W tym przypadku rezultaty są dość zbliżone do wartości minimalnych, otrzymanych w punkcie 5.1.3.1.

5.2 Graf ważony - przypadek z liczbą węzłów typu kernel określaną przez algorytm

Do zobrazowania działania naszej metody posłużyliśmy się grafem planarnym. Graf planarny, to graf, który można narysować na płaszczyźnie tak, aby krawędzie się nie przecinały. Grafy tego typu można uznać za dosyć dobre odwzorowanie części sieci logistycznych. Można przyjąć na przykład, że modelują sytuację, w której jedyne skrzyżowania dróg są w węzłach logistycznych. Oczywiście planarność wykorzystanych grafów została narzucona głównie do celów związanych z łatwiejszą prezentacją wyników i nie jest to warunek zastosowania opisywanych metod. W niniejszym artykule posłużyliśmy się grafem planarnym wygenerowanym przy użyciu programu ***yEd Graph Editor***. Przedstawiony graf jest dość mały, szczególnie w porównaniu z tymi, na których testowano metody dla grafów nieważonych, jednakże jest to spowodowane pragnieniem zamieszczenia czytelnych rysunków, a nie możliwościami obliczeniowymi metody. Graf o 64 wierzchołkach jest zbyt skomplikowany do optycznej analizy, jednakże można zauważyć, że węzły wybrane jako typu spoke są rzeczywiście najbliżej swoich hubów, natomiast węzły typu hub stanowią „rusztowanie" równomiernie pokrywające obszar grafu. Niestety połączenia między nimi są dość rzadkie i tworzą strukturę raczej drzewiastą niż grafu pełnego, ale wynika to z niezbyt dużej liczby krawędzi w grafie. Prawdopodobnie gdyby we wzorze (4.10) wprowadzić większy nacisk na część odpowiedzialną za połączenia między hubami, to wybranych zostałoby więcej wierzchołków, gęściej ze sobą połączonych, kosztem dalszych połączeń ze swoimi węzłami typu spoke.



Rys. 5.1. Graf planarny o 64 wierzchołkach.



Rys. 5.2. Struktura „hub and spoke”, odpowiadająca grafowi z rys. 5.1.

6. Wnioski

W pracy niniejszej zaproponowano kilka różnych metod rozwiązywania zadań związanych ze znajdowaniem rozbicia grafu na grupy silnie powiązanych ze sobą wierzchołków, tworzących jądro (kernel, węzły typu hub) grafu oraz wierzchołki peryferyjne typu shell (spoke). Przedstawiono możliwe klasy zastosowań tych metod.

Propozycje zaprezentowanego algorytmu genetycznego wykorzystują ona kilka adekwatnych, komputerowo wydajnych operacji uniwersalnych i specjalizowanych, kierując się tak złożonością obliczeniową jak i pamięciową. W przypadku tego algorytmu w mniejszym stopniu potrzebny jest ekspert w danej dziedzinie, a funkcja celu może być bardziej złożona. Niestety, zwykle płacimy za to zwiększonym nakładem obliczeń.

Prezentując różne wyniki obliczeniowe staraliśmy się pokazać jak wsparcie metod analitycznych w postaci środków graficznych pomaga prezentować rozwiązanie, a tym samym ocenić werbalnie ich wartość. Wizualizując rozwiązania na dostatecznie wczesnych etapach algorytmu można dzięki temu podpowiadać kolejne ulepszenia rozwiązania.

Bibliografia

Aho A.V., Hopcroft J.E., Ullman J.D. *Projektowanie i analiza algorytmów komputerowych*, PWN Poznań, 1982.

Altus S. S, Kroo I. M. , Gage P. J.*A Genetic Algorithm for Scheduling and Decomposition of Multidisciplinary Design Problems*, Journal of Mechanical Design, Vol. 118, No. 4, pp. 486-489, 1996.

Aussiello G., Crescenzi P., Gambosi G., Kann V., Marchetti-Spaccamela A., Protasi M.: Complexity and Approximation Springer, 1999.

Bagirov A. M.: *A new nonsmooth optimization algorithm for minimum sum-of-squares clustering problems*, EJofOR 170(2006) 578-595.

Berger B., Rompel J. *Efficient NC Algorithms for Set Cover with Applications to Learning and Geometry* Journal of Computer and System Sciences 49, 1994.

Błażewicz J. *Złożoność obliczeniowa problemów kombinatorycznych* WNT, 1988.

Benson S.J., Ye Y. (2000) *Approximating maximum stable set and minimum graph coloring problems with the positive semi-definite relaxation,* Computational complexity: The problem of approximation, Kluwer Academic Publishers.

Cormen T., Leiserson C., Rivest R. *Wprowadzenie do algorytmów,* WNT, Warszawa, 1997.

Hansen P., Mladenovic N., Urosevic D. *Variable neighborhood search for the maximum clique* The Fourth International Colloquium on Graphs and Optimisation (GO-IV)GO-IV International Colloquium on Graphs and Optimisation No4, Lòeche-les-Bains , SUISSE (20/08/2000) 2004, vol. 145, no 1 (28 ref.), pp. 117-125.

Hassin R., Khuller S. (1986) *z-Approximation*, Computational complexity: The problem of approximation, New York.

Hochbaum D. S. (Ed.) *Approximation Algorithms for NP-hard Problems*, PWS Publishing Company, 1997.

Hromkovic J. *Algorithmics for Hard Problems*, Springer, 2001.

Jukna S. (2001) *Extremal Combinatorics,* Springer-Verlag Berlin Heidelberg.

Kloks T., Kratsch D. (1998) *Listing all minimal separators of a graph,* SIAM J. COMPUT. Vol. 27, No. 3, pp. 605-613.

Korte B., Vygen J. *Combinatorial optimization, theory and algorithms*, Springer, 2000.

Kumlander D. (2007) *An Approach for the Maximum Clique Finding Problem Test Tool Software Engineering,* Software Engineering, SE 2007, Innsbruck, Austria.

Lenstra J.K.: Sequencing by Enumerative Methods, Mathematisch Centrum, Amsterdam (1997).

Lovasz L. *On the ratio of optimal integral and fractional covers* Discrete Mathematics 13 (1975).

Mażbic- Kulma B, Potrzebowski H., Stańczak J., Sęp K.(2008) *Evolutionary approach to solve hub-and-spoke problem using –αcliques,* Evolutionary Computation and Global Optimization, Prace naukowe PW, Warszawa, pp. 121-130.

McCormick W. T., Schweitzer P. J., White T. W. *Problem decomposition and data reorganization by a clustering technique*, Operations Res. 20, 1972, pp. 993-1009.

McCulley C., Bloebaum C. *A Genetic Tool for Optimal Design Sequencing in Complex Engineering Systems*, Structural Optimization, Vol. 12, No. 2-3, pp. 186-201, 1996.

O'Kelly M.E (1987) *A quadratic integer program for the location of interacting hub facilities,* European Journal of Operational Research 32, pp. 392-404.

O'Kelly M.E. and Bryan D. 2002: Interfacility interaction in models of hubs and spoke networks. Journal of Regional Science, 42 (1), 145-165.

Owsiński J.W. (1990) *On a new naturally indexed quick clustering method with a global objective function,* Applied Stochastic Models and Data Analysis,Vol. 6.

Potrzebowski H., Stańczak J., Sęp K. (2006a) *Heurystyczne i ewolucyjne metody znajdowania pokrycia grafu, korzystające z pojęcia alfa-kliki i innych ograniczeń* (in Polish), Badania operacyjne i systemowe 2006. Metody i techniki, Akademicka Oficyna Wydawnicza EXIT, Warszawa.

Potrzebowski H., J. Stańczak, Sęp K. (2006b) Evolutionary Algorithm to Find Graph Covering Subsets Using α-Cliques. Praca zbiorowa pod red. J.Arabasa: Evolutionary Computation And Global Optimization (2006), str. 351-358. Prace naukowe Politechniki Warszawskiej.

Potrzebowski H., Stańczak J., Sęp K.(2007) *Separable decomposition of graph using alpha-cliques,* in: Kurzyński. M., Puchała E., Woźniak M, Żołnierek A. (Eds.): Computer recognition systems 2, in: Advances in soft computing Springer-Verlag, Berlin-Heidelberg, pp. 386-393.

Potrzebowski H., Stańczak J., Sęp K. (2008) Evolutionary approach to solve hub-and-spoke problem using a-cliques, Evolutionary Computation and Global Optimization, Prace naukowe PW, Warszawa, pp. 121-130.

Protasi M. (2001) *Reactive local search for the maximum clique problem,* Algoritmica vol. 29, no 4. pp. 610-637.

Rogers J. L. *Reducing Design Cycle Time and Cost Thorough Process Resequencing*, International Conference on Engineering Design ICED 997, Tampere, Finland, 1997.

Sysło M. M., Deo N., Kowalik J. S. *Algorithms of discrete optimization*, Prentice-Hall, 1983.

Williamson D. *Lecture Notes on Approximation Algorithms* IBM Research Report RC 21409 02 17 1999.

Wilson R. J. (1996) *Introduction to graph theory* Addison Wesley Longman.

Yu T. L., Goldberg D. E., Yassine A., Yassine C. *A Genetic Algorithm Design Inspired by Organizational Theory*, Genetic and Evolutionary Computation Conference (GECCO) 2003, Chicago, Illinois, USA, Publ. Springer-Verlag, Heidelberg, Lecture Notes in Computer Science, Vol 2724/2003, pp. 1620-1621.

THE EVOLUTIONARY METHODS OF FINDING "KERNEL AND SHELL" STRUCTURES IN A GRAPH OF CONNECTIONS

Abstract: This paper describes the evolutionary methods of obtaining the "kernel & shell" structure in the graph of connections. The structure, proposed by the authors, is an extension of the well-known "hub & spoke" structure, widely used in communication and transport issues, enabling the transformation of unstructured connections graph to the form that assists in solving complex communication problems. The principle of this transformation is to identify in the graph the groups of closely connected nodes (hubs), forming the structure of the "kernel" and associated peripheral nodes, forming a "shell". Since this task is computationally complex, an evolutionary algorithm is used for the purpose in some variants, depending on the application needs and the problem under consideration.

Keywords: transportation, connectivity graph, "hub & spoke", "kernel & shell", α-clique, evolutionary algorithm.